Um Algoritmo de Diagnóstico Distribuído para Redes de Topologia Dinâmica

Leandro Pacheco de Sousa

[1]Instituto de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

leandrops@inf.ufpr.br

Abstract. *This work aims to design an algorithm for distributed failure diagnosis on networks with dinamic topologies. The algorithm will be based upon the algorithm presented on [Sousa 2006]. The modification of the algorithm can be divided in two steps. First, the algorithm must be modified to work correctly in the case of network partitioning. Second, the algorithm must be extended to work with the insertion and removal of nodes during its execution. The latter is a considerably more complex task, since one of the premises of the base algorithm is that the network topology is static and every node knows it when the execution starts. When the algorithm is defined, it will be simulated and tested using the ns-2 simulator.*

Resumo. *Este trabalho tem como objetivo desenvolver um algoritmo para diagnóstico de falhas em redes de topologia dinâmica. Este algoritmo será baseado no algoritmo apresentado em [Sousa 2006]. A modificação do algoritmo original pode ser dividida em dois passos. Primeiro, o algoritmo precisa ser modificado para funcionar corretamente no caso de particionamento na rede. Segundo, o algoritmo precisa ser extendido para suportar a inserção e remoção de nodos durante sua execução. Esta última é uma tarefa consideravelmente mais complexa, pois uma das premissas do algoritmo original é que a topologia é estática e todo nodo tem conhecimento da mesma quando a execução do algoritmo inicia. Após a definição do algoritmo, ele será implementado e testado no simulador ns-2.*

1. Introdução

Os Sistemas de gerência de redes permitem controlar e monitorar os elementos de uma rede. Uma função destes sistemas é a gerência de falhas. Os algoritmos de diagnóstico distribuído podem ser utilizados como base dos sistemas de gerência de falhas. O algoritmo *Distributed Network Reachability* [Duarte and Weber 2003, Weber et al. 2006], ou simplesmente DNR, é um algoritmo de diagnóstico em distribuído para redes de topologia arbitrária. Neste algoritmo, um enlace pode assumir 3 estados, falho, não-falho e inatingível, que é quando este não é alcançável no estado atual da rede. Nodos podem assumir apenas dois estados, sem-falha ou inatingível. Isto ocorre devido à ambiguidade das falhas em redes de topologia arbitrária. É impossível determinar se um nodo está falho ou não se todos os seus enlaces não estiverem respondendo. O algoritmo consiste de três fases. Na fase de testes, são descobertas falhas e recuperações de nodos ou encales, o que chamamos de eventos. Com a descoberta de um novo evento, é iniciada a fase de

disseminação. Na fase de disseminação a informação sobre o novo evento é distribuída para toda a rede. A ultima fase é chamada fase de diagnóstico. A fase de diagnóstico pode ser iniciada por qualquer nodo sem-falha, a qualquer momento. Nesta fase é feito o cálculo de alcançabilidade da rede através do uso de um algoritmo de conectividade em grafos. Uma das premissas do algoritmo DNR é o conhecimento prévio da topologia da rede. A topologia é estática e todos os nodos já conhecem a mesma quando o algoritmo se inicia.

Em [Sousa 2006] é apresentada a implementação de uma variação do algoritmo DNR. Esta implementação é feita com o uso do simulador *ns-2*. As diferenças entre o algoritmo implementado e o DNR estão na fase de disseminação. A disseminação no algoritmo implementado é feita com o uso de uma técnica de inundação. Outra diferença do está no fato do mesmo não tratar do caso de particionamento e posterior recuperação da rede.

2. Problema

O algoritmo de diagnóstico proposto em [Sousa 2006] possui duas limitações. A primeira é que este não leva em conta o caso de particionamento na rede. A segunda limitação é que o algoritmo precisa de conhecimento prévio sobre a topologia da rede. Quando o algoritmo se inicia, todos os nodos possuem uma tabela contendo a topologia da rede e esta topologia se mantém estática durante a execução do algoritmo. Este trabalho tem como objetivo modificar o algoritmo para remover estas duas limitações.

2.1. Particionamentos na Rede

Em redes de topologia arbitrária, falhas de nodos ou enlaces podem causar o particionamento na rede. Isto pode gerar problemas na execução do algoritmo em questão. Durante o particionamento cada componente conexo da rede continua executando o algoritmo normalmente. Quando um evento de recuperação que reconecte estes componentes ocorrer, podem haver nodos com informações desatualizadas. Para resolver este problema, as informações contidas nos nodos de cada componente conexo precisam ser repassadas para o resto da rede. Existe uma maneira simples de resolver isso. Quando um nodo detecta uma recuperação em um enlace qualquer, ele deve enviar por este enlace toda informação que ele possui sobre o estado da rede. Isto deve ocorrer para ambos os nodos do enlace em questão, para que a informação de ambos os lados seja repassada para toda a rede. Esta solução pode causar o envio de muitas mensagens em caso de redes muito grandes ou no caso de muitas falhas e recuperações. Uma maneira de diminuir o número de mensagens enviadas neste caso seria com a detecção do caso de particionamento. Assim, o reenvio de toda informação dos nodos seria feita somente na recuperação de um particionamento. Esta solução pode não ser viável. É preciso verificar se podem surgir casos onde esta não funcione corretamente. De qualquer maneira, a modificação provavelmente estará restrita à fase de disseminação.

2.2. Topologia Dinâmica

Tanto no algoritmo proposto em [Sousa 2006] quanto no algoritmo DNR, existe a premissa de que todos os nodos possuem conhecimento sobre a topologia da rede quando o algoritmo inicia. Esta topologia se mantém estática durante toda a execução do algoritmo.

A segunda modificação proposta ao algoritmo em questão é adicionar o suporte à topologias dinâmicas. Durante a execução do algoritmo, poderia ocorrer a inserção de novos nodos ou novos enlaces e também a remoção de nodos ou enlaces existentes. Esta é uma modificação que parece ser relativamente complexa, pois haverão modificações nas três fases do algoritmo, testes, disseminação e diagnóstico.

Para a inserção de novos nodos, é preciso adicionar ao algoritmo uma maneira de um novo nodo enviar a informação de sua entrada no grupo para todos os seus vizinhos. Esta informação precisa então ser distribuída para o resto da rede. Para o caso de um novo enlace, os nodos das extremidades do enlace devem iniciar a disseminação na rede. Isto já pode gerar alguns problemas que devem ser tratados. Uma inserção pode ocorrer durante um particionamento na rede. Nodos de outro componente conexo não ficarão sabendo da inserção do novo nodo ou enlace. A solução para este problema talvez esteja relacionada à primeira modificação proposta. Quando ocorrer a detecção de uma recuperação de enlace, os nodos das extremidades enviam informação sobre a estrutura atual da rede. É preciso tomar cuidado para que a inserção dos novos nodos ou enlaces sejam levadas em conta em todas as fases do algoritmo.

Para a remoção de nodos, uma maneira de tratar o problema seria com o nodo que irá sair do grupo enviando uma mensagem para seus vizinhos informando sobre sua saída. Talvez seja necessário o uso de *acks* para esta informação, já que pode existir o caso onde todos os enlaces deste nodo falhem simultanêamente durante o envio da informação. Na remoção de enlaces, uma das pontas precisa iniciar o processo de remoção, avisando para todos os vizinhos sobre a remoção, inclusive o vizinho ligado ao enlace em questão. Em ambos os casos, podem surgir vários problemas. Um deles é o caso da remoção do nodo ou enlace provocar o particionamento da rede. Neste caso, se um dos componentes conexos da rede não receber a informação durante a remoção, ele jamais irá receber esta informação. Na figura 1 estes casos são ilustrados. Talvez seja interessante que a remoção de um nodo seja implementada como uma remoção de enlaces. Assim, se um nodo quiser sair da rede, ele primeiro inicia a remoção de todos os seus enlaces. Quando isto ocorrer, ele pode se desligar da rede. Da mesma maneira que com inserções, é preciso considerar uma remoção em todas as fases do algoritmo, pois podem haver consequências não previstas.

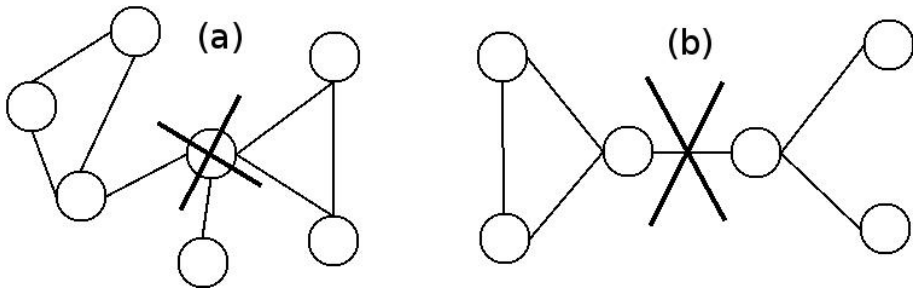

Figure 1. Remoção de nodo(a) e enlace(b) causando particionamento

3. Implementação

Em [Sousa 2006] além da definição do algoritmo, também é descrita uma implementação do mesmo no simulador *ns-2*. O *ns-2* é um simulador de eventos discretos bastante utilizado na área de pesquisa em redes de computadores. A implementação descrita propõe

a criação de um novo agente e uma nova aplicação para o *ns-2*. Estes são chamados respectivamente de *DNRAgent* e *DNRApp*. Cada *DNRAgent* está ligado a um nodo e possui uma conexão com outro *DNRAgent* ligado a outro nodo. Esta conexão forma o canal de comunicação entre dois nodos, caracterizando um enlace. Desta maneira, o número de agentes ligados à um nodo qualquer é igual ao número de nodos vizinhos. Os agentes são utilizados apenas para envio e recebimento de mensagens do algoritmo. As aplicações *DNRApp* são os componentes que executam o algoritmo. Cada nodo possui exatamente uma *DNRApp*, que está ligada à todos os agentes do nodo. Cada aplicação executa o algoritmo de maneira independente das outras e possui informação sobre a topologia e o estado de toda a rede. A figura 2 mostra a configuração do simulador para uma topologia exemplo.

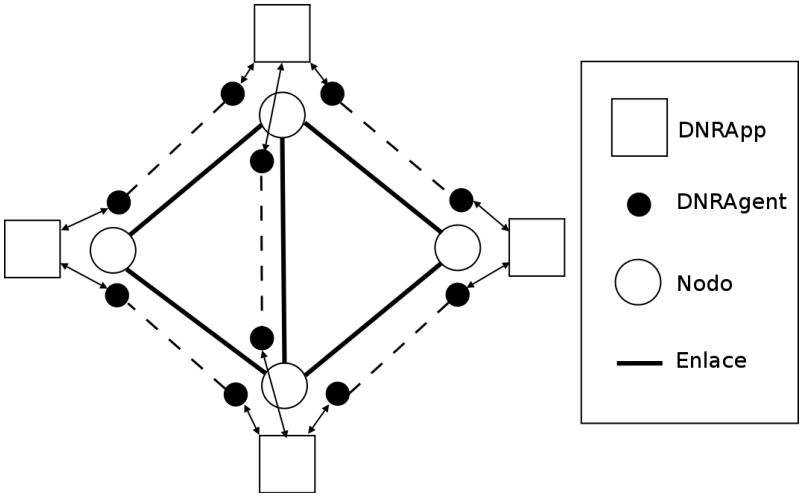

Figure 2. Configuração de nodos, agentes e aplicações para uma topologia exemplo

Este trabalho pretende acrescentar as modificações propostas anteriormente à implementação existente. Para adicionar as modificações, talvez serão necessárias algumas reestruturações na maneira como o algoritmo foi implementado. Será preciso estudar uma maneira de simular a inserção e remoção de nodos e enlaces, modificando o agente e aplicação propostos para suportar os dois casos.

Terminada a implementação, serão realizados testes para verificar a corretude do algoritmo em alguns casos pré-definidos e possivelmente algumas topologias geradas aleatoriamente. Os testes também serão utilizados para obtenção de alguns dados relacionados à execução, como número de mensagens geradas para disseminação de um novo evento e a latência do algoritmo para informar toda a rede de um dado evento. A aplicação *nam* será utilizada para visualização do comportamento do algoritmo.

4. Conclusão

Apresentamos neste trabalho duas propostas de modificação ao algoritmo de diagnóstico distribuído descrito em [Sousa 2006]. A primeira modificação teve por objetivo tornar o algoritmo tolerante a particionamentos na rede. A segunda modificação foi proposta para que o algoritmo pudesse executar em redes de topologia dinâmica. Também propusemos a implementação do algoritmo modificado no simulador *ns-2*, onde serão realizados testes para verificar a corretude e a eficiência do algoritmo.

5. Cronograma de Atividades

Mês	Abril				Maio				Junho			
Semana	1	2	3	4	1	2	3	4	1	2	3	4
Revisão da implementação no ns-2		X	X									
Modificar o algoritmo para suportar o particionamento				X								
Implementação e teste da 1a modificação					X							
Estudar inserção/remoção de novos nodos no algoritmo(possível reestruturação)						X						
Modificar o algoritmo para aceitar inserção e remoção de nodos						X	X	X				
Implementação do algoritmo e realização de testes							X	X	X			
Elaboração do relatório final									X	X	X	
Elaboração da apresentação											X	

References

Duarte, E. P. and Weber, A. (2003). A distributed network connectivity algorithm. In *Proc. IEEE/ISADS'03*, Pisa.

Sousa, L. P. (2006). Simulação de um algoritmo de diagnóstico baseado em token-test com disseminação por inundação. Monografia apresentada ao Curso de Bacharelado em Ciência da Computação.

Weber, A., Duarte, E. P., and Fonseca, K. V. O. (2006). An optimal test assignment for monitoring general topology networks. In *The 7th IEEE Latin Amerincan Test Workshop (IEEE LATW'2006)*, Buenos Aires, Argentina.