Potência do Sinal de Recepção como Suporte à Detecção de Mobilidade em Detectores de Defeitos

Miguel A. Baggio, Raul Ceretta Nunes, Marcia Pasin, Antônio Rodrigo D. de Vit

Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal de Santa Maria (UFSM)
Av. Roraima 1000 - Cidade Universitária - 97105-900 – RS – Brasil

{baggio, ceretta, marcia}@inf.ufsm.br; rodrigodevit@smail.ufsm.br

Abstract. *Failure detector is a building block component in reliable distributed systems and its design depends strongly on the model of the distributed system. This dependence has demanded new algorithms to address nodes movement on mobile ad hoc networks (MANETs). This paper presents a new unreliable gossip-based failure detector that differentiates faulty and mobile nodes. Our approach explores the information about signal power of received messages mapped into a little locality region history. The experimental results have shown quality of service improvements when compared with the traditional gossiping algorithm.*

Resumo. *Detector de defeitos é um componente essencial na construção de sistemas distribuídos confiáveis e seu projeto depende fortemente do modelo de sistema distribuído, o que tem demandado soluções para tratar a movimentação de nós em redes móveis ad hoc (MANETs). Este trabalho apresenta um detector de defeitos assíncrono não-confiável baseado em fofoca que diferencia nós defeituosos e móveis através da manutenção de informações sobre a potência do sinal de recepção nos nós do sistema mapeadas em um pequeno histórico de regiões. As avaliações demonstram melhoras na qualidade de serviço do detector quando comparadas com o algoritmo de fofoca tradicional.*

1. Introdução

Algoritmos para detecção de defeitos, um bloco de construção básico para a construção de sistemas distribuídos confiáveis, têm sido amplamente estudados para operação em redes móveis sem fio [Hutle 2004][Friedman e Tcharny 2005][Sridhar 2006][Sens *et al*. 2008] [Mahapatro e Khilar 2013]. Entretanto, ainda persiste o desafio de diferenciar precisamente a mobilidade de um nó de um estado defeituoso.

Especialmente em MANETs (*Mobile Ad hoc NETworks*) [Loo *et al*. 2012], uma rede auto-configurável (sem uma administração centralizada) em que os nós caracterizam-se por serem móveis e possuírem conexões dinâmicas, diferenciar uma desconexão por movimentação de uma desconexão por falha do nó exige um algoritmo que permita classificar um nó como móvel, descartando assim a possibilidade de falha. Em MANETS, os nós comunicam-se com seus vizinhos por mensagens *broadcast* via rádio, limitado ao alcance de transmissão de seus rádios. Para um detector de defeitos a característica de mobilidade dos nós e a limitação no alcance de transmissão podem

ocasionar falsas suspeitas (imprecisão), dado que um nó p_i pode suspeitar erroneamente que um nó p_j esteja falho quando, de fato, p_j pode apenas ter se movimentado para uma região onde seu alcance de transmissão não alcança p_i.

De acordo com Gracioli e Nunes (2007), para não depender de hierarquia ou topologia fixa, grande parte das estratégias de detecção de defeitos para redes móveis sem fio tem como base o algoritmo *Gossip* [Renesse *et al.* 1998]. O *Gossip* é um algoritmo epidêmico baseado no fenômeno social chamado fofoca, onde um nó difunde (por *broadcast* ou *multicast*) para seus vizinhos locais (dentro do seu alcance de transmissão) informação sobre um grupo de nós vizinhos (locais ou remotos). Neste algoritmo, quando um nó detector de defeitos fica muito tempo sem receber notícias sobre um determinado nó (equivalente a um tempo nomeado *Tcleanup*), o detector suspeita do nó. O ponto chave do algoritmo é que um nó conhece seus vizinhos ao longo do tempo através de mensagens que passam de vizinhos para vizinhos.

Como a estratégica epidêmica por si só não permite diferenciar nós defeituosos de nós móveis, com base nela foram exploradas técnicas de contagem de *hops* [Friedman e Tcharny, 2005], de determinação da variação do atraso de comunicação [Hutle, 2004], de inclusão de novos *rounds* de comunicação [Zia *et al.*, 2009] e de difusão de listas de informações [Sridhar, 2006]. Adicionalmente, Sens *et al*. (2008) propuseram uma técnica que independe de temporizador e que habilita a detecção apenas com base em informações sobre falhas. Diferentemente dos demais, este artigo explora a característica de variabilidade de potência nas transmissões de uma rede MANET [Gomez 2007] e propõe uma nova técnica de detecção de defeitos para redes móveis sem fio: o uso da medida de potência do sinal de recepção como elemento classificador da mobilidade de um nó. Classificando os nós em regiões, a técnica proposta permite identificar a movimentação de um nó e gerar uma informação de previsão de movimentação do nó, a qual é utilizada pelo detector de defeitos para se ajustar à mobilidade e melhorar a qualidade da detecção de defeitos. A solução proposta foi avaliada em um ambiente de simulação para MANETs e demonstrou melhora na qualidade de serviço do detector.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta o modelo de sistema e discute detectores de defeitos. A Seção 3 apresenta definições e conceitos usados ao longo deste artigo. A Seção 4 detalha a proposta de detector de defeitos que usa a potência do sinal de recepção como parâmetro de classificação no suporte à mobilidade. A Seção 5 apresenta os resultados práticos da simulação do algoritmo. A Seção 6 apresenta os trabalhos relacionados e a Seção 7 conclui o trabalho.

2. Modelo de Sistema e Detectores de defeitos

Esta seção apresenta o modelo de sistema distribuído considerado e uma revisão sobre os detectores de defeitos.

2.1 Modelo de Sistema

O sistema distribuído considerado neste trabalho é assíncrono e composto por um conjunto finito de nós móveis $\prod = \{p_1, \ldots, p_n\}$, onde $n > 1$. Cada nó tem sua própria memória, unidade de processamento, relógio local e processos em execução. Não há relógio global. Os nós se comunicam através de troca de mensagens, usando difusão

(*broadcast*) via rádio com alcance de transmissão finito e igual para todos os nós. A potência do sinal de recepção é variável dentro do alcance de transmissão. Uma mensagem *m* enviada por um nó p_i somente é recebida por outro nó p_j que esteja dentro do seu alcance de transmissão quando *m* for enviada e a potência do sinal de recepção da mensagem *m* puder ser identificada por p_j. Todo nó móvel p_i percorre, num tempo finito, pelo menos uma trajetória *T* tal que ao longo de *T* o nó p_i entra no alcance de transmissão de pelo menos um nó p_j e recebe uma mensagem *m* de p_j, e pelo menos um nó p_j recebe uma mensagem de p_i. A topologia de rede é dinâmica e os canais de comunicação são bidirecionais e confiáveis (não alteram, não criam e não perdem mensagens). Os nós são auto-organizáveis e adaptam-se de forma autônoma a falhas do tipo *crash*. Para controlar falhas, cada nó no sistema distribuído executa um serviço de detecção de defeitos.

2.2 Detector de defeitos

Dada a natureza assíncrona do sistema distribuído, o conceito de detectores de defeitos não confiáveis foi introduzido por Chandra e Toueg (1996) para contornar o problema da impossibilidade de resolver o consenso num sistema assíncrono sujeito a falhas [Fischer *et al*. 1985]. A ideia básica é encapsular o problema no detector de defeitos e fornecer uma noção de estado para a aplicação. O estado pode ser expresso por uma lista de nós suspeitos de estarem falhos [Chandra e Toueg, 1996], uma lista de nós confiáveis [Aguilera *et al*. 1998] ou uma lista de contadores de mensagens [Aguilera *et al*. 1997]. Normalmente o estado que é informado à aplicação por um detector de defeitos corresponde à percepção do módulo de detecção local e um nó p_j é dito *suspeito* quando um detector p_i não recebe uma mensagem do nó p_j dentro de um determinado intervalo de tempo.

Há diferentes estratégias de comunicação para construir a noção de estado no detector [Felber *et al*. 1999]. Em redes cabeadas, as mais tradicionais são: *push*, estratégia baseada no envio periódico de mensagens de estado corrente (*I_am_alive!*) ou *heartbeat* [Aguilera *et al*. 1997]); e *pull*, estratégia baseada no envio periódico de mensagens de solicitação de estado (*Are_you_alive?*) com consequente confirmação (*Yes_I_am!*). Em redes móveis sem fio, incluindo MANETs, a mais tradicional é a estratégia baseada em fofoca [Gracioli e Nunes 2007][Friedman *et al*. 2007], a qual é baseada no envio periódico de mensagens *gossip* que são repassadas de vizinhos locais para vizinhos remotos. Normalmente uma mensagem *gossip* carrega uma lista de identificadores e contadores de *heartbeat*.

Em MANETs, independente do uso de um algoritmo epidêmico, que dissemina a informação de nó para nó através de uma estratégia eficiente e escalável baseada em fofoca ou boatos, é preciso distinguir um comportamento defeituoso de uma movimentação do nó na rede.

3. Definições e Conceitos

Esta seção apresenta um conjunto de definições e conceitos relacionadas ao algoritmo de detecção de defeitos proposto na Seção 4, visando facilitar sua compreensão.

Definição 1. ***Alcance de transmissão.*** O alcance de transmissão (*range*) é o limite máximo de distância que uma mensagem emitida por um nó p_i pode chegar e é igual para todo $p_i \in \prod$.

Definição 2. ***Grupos de comunicação.*** O sistema pode ser representado por vários grupos de comunicação GC_i, formados por nós p_i e seus vizinhos locais em um dado instante de tempo t. Um nó p_j é dito vizinho local de p_i, se p_j pertence ao alcance de transmissão de p_i. Uma mensagem m enviada pelo nó p_i é recebida por todos os vizinhos corretos (não falhos) de p_i pertencentes a GC_i.

Das definições 1 e 2 tem-se que todos os nós do sistema têm o mesmo *range*, logo:

$\forall \{p_i, p_j\} \in \prod$: se $p_i \subset range_j$ então $p_j \subset range_i$

$\forall p_i \in \prod$: se $p_j \in GC_i$ então $p_j \subset range_i$

Definição 3. ***Regiões de cobertura.*** Assumindo que o alcance de transmissão de uma mensagem numa MANET é proporcional à potência do sinal de transmissão, a área de cobertura de um nó p_i (área dentro do alcance de transmissão) pode ser dividida em regiões de cobertura (vide Figura 1), onde cada região corresponde a um nível de potência do sinal de recepção (quanto mais longe do nó menor a potência).

Das definições 1 e 3, para uma dada região k, tem-se que

$\forall \{p_i, p_j\} \in \prod$: se $p_i \subset região_j^k$ então $p_j \subset região_i^k$.

A Figura 1 ilustra o *range* de um nó p_i e suas respectivas regiões de cobertura, as quais são distinguidas pela potência do sinal de recepção. Desta forma, através da análise da potência de recepção de cada mensagem recebida, p_i pode então localizar um nó p_j dentro de uma de suas regiões de cobertura.

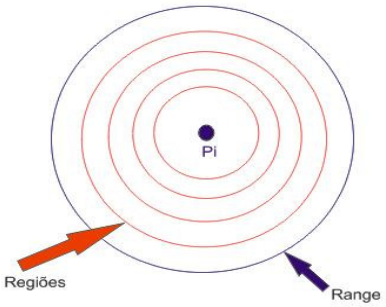
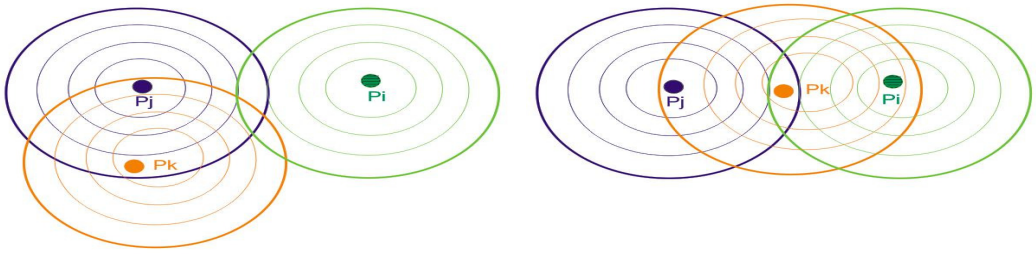


Figura 1. Regiões de Cobertura

Cada nó p_i pode se mover arbitrariamente na rede, mudando a configuração de seu grupo de comunicação GC_i. Desta forma, um nó p_i pode sair de um GC_k de p_k e entrar no GC_j de um nó p_j. Também pode ocorrer sobreposição de grupos de comunicação: um nó p_i pode pertencer aos grupos de comunicação dos nós p_j e p_k simultaneamente. A Figura 2 ilustra dois cenários com diferentes grupos de comunicação. Na Figura 2(*a*) p_k pertence a GC_j e não pertence ao GC_i. Na Figura 2(*b*) p_k pertence ao GC_j e ao GC_i. Em ambos os casos p_k também pertence ao GC_k.



(a) (b)

Figura 2. Cenários de Grupos de Comunicação

Definição 4. *Histórico de movimentação*. Cada nó p_i mantém um histórico H_i^j para todo p_j pertencente a GC_i, no qual são mantidas as informações sobre a região que cada nó vizinho p_j está. O histórico é atualizado em um intervalo de tempo finito e a cada atualização a informação mais atual sobre a região de p_j prevalece.

Definição 5. *Lista de suspeitos.* Cada nó p_i mantém uma lista de suspeitos SL_i. Um nó p_j é adicionado a SL_i se e somente se p_i não detecta movimentação de p_j nem recebe nenhuma mensagem de p_j dentro de um intervalo de tempo finito.

Definição 6. *Lista de mobilidade*. Cada nó p_i mantém uma lista de mobilidade ML_i. Um nó p_j é adicionado a ML_i se e somente se p_i detecta movimentação de p_j, a qual é baseada na análise das regiões de cobertura observadas a cada mensagem recebida.

Definição 7. *Lista de vizinhos.* Cada nó p_i mantém uma lista de vizinhos NBL_i, que corresponde a um conjunto de nós sobre os quais p_i recebe mensagens ou informações e acredita estarem corretos. Num dado instante de tempo a lista de vizinhos NBL_i pode conter: nós vizinhos locais (pertencentes ao GC_i); nós que se deslocaram para outra vizinhança (não mais pertencentes ao GC_i); e nós que nunca pertenceram ao GC_i.

4. Detector de defeitos com Suporte à Mobilidade

Esta seção apresenta a proposta de detector de defeitos não confiável para MANETs baseado na potência do sinal de recepção das mensagens. A Seção 4.1 detalha a adaptação do mecanismo básico de detecção do algoritmo Gossip, a Seção 4.2 detalha as estruturas de dados utilizadas e a Seção 4.3 detalha o algoritmo proposto. Finalmente a Seção 4.4 apresenta as propriedades e a prova de correção do algoritmo.

4.1 Adaptação do mecanismo básico para detecção de defeitos do algoritmo Gossip

Para construção da lógica de detecção de defeitos assume-se que todo nó monitora todos os outros nós do sistema através do envio de mensagens *gossip* de acordo com o protocolo *Gossip* básico [Renesse *et al.* 1998]. Porém, o mecanismo de detecção de defeitos foi adaptado como segue.

No *Gossip* básico a detecção de defeitos segue o seguinte algoritmo: a cada intervalo *Tgossip*, um nó escolhe aleatoriamente um ou mais vizinhos para enviar a mensagem *gossip*; no recebimento da mensagem, a lista de identificadores é unida com a lista que o receptor possui e o maior valor do contador de *heartbeats* de cada nó é mantido na lista final; cada nó mantém o instante do último incremento do contador de *heartbeat*; se o contador não for incrementado em um intervalo de *Tfail* unidades de

tempo então o nó é considerado suspeito, mas colocado na lista de suspeitos apenas após *Tcleanup* unidades de tempo; o *Tfail* é escolhido de acordo com a probabilidade de ocorrer uma detecção de defeito errônea (*Pmistake*) e o *Tcleanup* é escolhido de modo a fazer com que a probabilidade de receber uma mensagem *gossip* que evite uma suspeita errônea seja menor que *Pcleanup*.

É possível simplificar o protocolo *Gossip* permitindo que nós sejam removidos da lista depois de *Tcleanup* sem a necessidade de verificação do *Tfail* [Burns et al. 1999]. O ponto chave para esta simplificação é que o *Tcleanup* pode ser definido como múltiplo do *Tgossip* e precisa corresponder ao tempo necessário para as informações alcançarem outros nós dentro do tempo limite *Tcleanup* [Subramaniyan et al. 2005].

Neste trabalho, simplificamos as escolhas dos parâmetros do algoritmo *Gossip* fazendo *Tcleanup* ser ajustado para *Cfail* × *Tgossip,* sendo *Cfail* o número de intervalos *Tgossip* que devem ser aguardados até que se confirme a suspeita de falha em um dado nó *p*. Como *Cfail* numa MANET depende do padrão de movimentação de cada nó, existe um *Cfail* para cada nó $p \in \prod$.

4.2 Estruturas de dados para suporte à mobilidade

Cada nó p_i pertencente a $\prod$ mantêm uma lista de suspeitos SL_i (Definição 5), uma lista de mobilidade ML_i (Definição 6) e uma lista de vizinhos NBL_i (Definição 7). O nó p_i mantém as informações dos seus vizinhos em uma estrutura de dados NB_i, onde para cada vizinho p_j é mantido:

- H_i^j - histórico de movimentação do nó p_j ;
- TS_j - *timestamp* da última informação recebida do nó p_j ;
- $Hcount_j$ - contador de *heartbeats* referente às mensagens *gossip* do nó p_j ;
- $Cfail_j$ - contador de intervalos *Tgossip* que devem ser aguardados para sobrepor o tempo de desconexão local do nó p_j dado seu padrão de movimentação (trajetória de movimentação); e
- $Rpot_j$ - informação sobre a região (localização) de p_j na região de cobertura de p_i.

4.3 Algoritmo de detecção de defeitos com suporte à mobilidade

Para detalhar o funcionamento do algoritmo proposto esta subseção está dividida de acordo com as tarefas concorrentes do algoritmo, as quais se dividem em: 1 - envio de mensagens; 2 - recebimento de mensagens; e 3 - verificação de estado. O algoritmo completo é apresentado na Figura 3. No início, as estruturas de dados são ajustadas para que um nó p_i suspeite de todos os outros nós, exceto de si próprio.

Tarefa 1: envio de mensagens

Todo nó que executa o algoritmo de detecção de defeitos com suporte à mobilidade envia uma mensagem *gossip* por *broadcast* de rádio a cada intervalo *Tgossip*. Esta mensagem contém a sua lista de vizinhos (NBL_i) e a sua lista de contadores de *heartbeats* (*Hcount*). Este mecanismo faz com que a cada intervalo *Tgossip* um nó p_j dentro de uma das regiões de cobertura do nó p_i receba uma mensagem de p_i com informações sobre o nó p_i e seus vizinhos não suspeitos

(pertencentes ou não ao grupo de cobertura de p_i). Esta tarefa corresponde às linhas 7 a 9 do algoritmo de detecção com suporte à mobilidade ilustrado na Figura 3.

Note que o envio de NBL_i permite que a informação sobre um nó p_k vizinho de p_i e fora da vizinhança local de p_j, ou seja, fora da alcance de transmissão de p_j, possa ser difundida para p_j. Deste modo o nó p_j poderá passar a reconhecer o nó p_k como um de seus vizinhos (incluindo p_k na sua lista de vizinhos).

Tarefa 2: recebimento de mensagens

A cada mensagem *gossip* recebida de um nó p_j , o nó p_i verifica a potência do sinal de recepção referente a mensagem e atualiza $Rpot_j$ (linha 13). A região $Rpot_j$ onde p_j se localiza é então inserida no histórico H_i^j (linha 14). Se duas ou mais mensagens *gossip* forem recebidas de p_j num mesmo intervalo de verificação *Tcleanup*, apenas a informação mais recente será mantida no histórico.

Se p_j pertence a ML_i, então o nó p_j é adicionado em NBL_i e removido da lista ML_i (linhas 15 a 17). Por outro lado, se o nó p_j pertence a SL_i, é verificado quantos *Tgossip* o nó ficou como suspeito e $Cfail_j$ é atualizado (linha 20). Este procedimento permite que o nó p_i mantenha informação sobre o padrão de mobilidade do nó p_j e, num tempo finito, consiga evitar que ele seja novamente adicionado de maneira errada como suspeito. Note que p_j pode estar fora do alcance de transmissão mas não estar defeituoso. O nó é então adicionado novamente à lista NBL_i e removido da lista SL_i (linhas 21 e 22).

O nó p_i também atualiza informações gerais referentes a todos os nós não suspeitos p_k pertencentes a NBL_j (linhas de 24 a 35). O contador de *heartbeats* é atualizado para o maior valor mantido por p_i e p_j, e, em caso de atualização, o *timestamp* também é atualizado. Se p_k pertence a ML_i ou a SL_i então o nó p_k é adicionado em NBL_i e removido da ML_i ou SL_i, respectivamente. Se o nó p_k pertence a SL_i, o número de intervalos *Tgossip* que p_i ficou sem receber mensagens de p_k é computado e esse valor é atribuído a $Cfail_k$. Deste modo, um nó fora de GC_i pode se manter como não suspeito (em NBL_i).

Tarefa 3: verificação de estado

O mecanismo de verificação de estado serve para atualizar a visão local de um nó p_i sobre sua vizinhança ($NBL_i \cup ML_i$), incluindo os nós considerados móveis. Sua tarefa está dividida em duas fases: quando $p_j \in NBL_i$ e quando $p_j \in ML_i$.

Quando p_j pertence a lista de vizinhos NBL_i (linhas 39 a 48) o algoritmo verifica se o nó p_i recebeu informação dos nós vizinhos em um intervalo de tempo esperado ($Cfail_j$ * *TGossip*). Se o tempo de espera por uma mensagem do nó p_j não for superior ao esperado, o histórico de p_j é atualizado com a região referente ao recebimento da última mensagem de p_j (linha 48). Se o tempo de espera limite foi ultrapassado (teste na linha 40) o nó p_i atualiza o histórico H_i^j com uma informação vazia [-] (linha 41), indicando que p_i não recebeu mensagem ou informação de p_j no intervalo esperado. Além disto, quando extrapolado o tempo limite, se H_i^j [t-1] for igual ao H_i^j [t-2] e o contador de intervalos $Cfail_j = 0$, tem-se uma situação em que p_j não teve movimentação prévia, indicando uma potencial falha do nó. Logo, o nó é imediatamente adicionado à lista de suspeitos SL_i e removido da lista de vizinhos NBL_i (linhas 43 e 44). Por outro

lado, o algoritmo detecta uma movimentação de uma região para outra no intervalo de tempo [t-1, t-2] se H_i^j [t-1] ≠ H_i^j [t-2]. Deste modo, o nó é adicionado na lista de mobilidade ML_i e removido da sua lista de vizinhos NBL_i (linhas 46 e 47). Note que os nós em ML_i continuam na vizinhança de p_i, mas em movimento.

```
1.   H ← 0 ; Cfail ← 0 ; Rpot ← 0 ;
2.   NBL ← { };              //Lista de vizinhos
3.   ML ← { };               //Lista de Mobilidade
4.   SL ← { p_i, ..., p_n };  //Lista de Suspeitos
5.
6.   Task 1: Difusão
7.   a cada Tgossip faça
8.       Hcount_i = Hcount_i +1;
9.       broadcast (NBL_i, Hcount);
10.
11.  Task 2: Recebimento de Mensagem
12.  a cada mensagem gossip de p_j com (NBL_j, Hcount) faça
13.      atualize Rpot_j ;      //Rpot é a região correspondente a potência do sinal da msg
14.      atualiza H_i^j com Rpot_j;
15.      Se p_j ∈ ML_i então
16.              adiciona p_j em NBL_i;
17.              remove p_j de ML_i;
18.
19.      Se p_j ∈ SL_i então
20.              Cfail_j = (TL_i - TS_j ) / Tgossip;  // TL_i = hora atual de p_i e TS_j = timestamp_j
21.              adiciona p_j em NBL_i;
22.              remove p_j de SL_i;
23.
24.      ∀ p_k ∈ NBL_j com i ≠ k faça
25.              atualiza Hcount_i^k com Max(Hcount_i^k, Hcount_j^k)
26.              Se Hcount_j^k > Hcount_i^k então TS_k = TL_i;
27.
28.              Se p_k ∈ ML_i então faça
29.                      adiciona p_k em NBL_i;
30.                      remove p_k de ML_i;
31.
32.              Se p_k ∈ SL_i então faça
33.                      Cfail_k = (TL_i - TS_k ) / Tgossip;
34.                      adiciona p_k em NBL_i;
35.                      remove p_k de SL_i;
36.
37.  Task 3: Verificação de Estado
38.  a cada TCleanup faça
39.      ∀ p_j ∈ NBL_i com i ≠ j faça
40.              se TL_i - TS_j > Cfail_j * TGossip então
41.                      atualiza H_i^j com '-';
42.                      se H_i^j [t-1] = H_i^j [t-2] e Cfail_j = 0 então
43.                              adiciona p_j em SL_i;
44.                              remove p_j de NBL_i;
45.                      senão
46.                              adiciona p_j em ML_i;
47.                              remove p_j de NBL_i;
48.              senão atualize H_i^j (Rpot_j);
49.      ∀ p_j ∈ ML_i com i ≠ j faça
50.              se TL_i - TS_j > Tgossip + (Cfail_j * Tgossip) então
51.                      adiciona p_j em SL_i;
52.                      remove p_j de ML_i;
```

Figura 3. Algoritmo de detecção de defeitos com suporte à mobilidade

Quando p_j pertence a ML_i (linhas 49-52) é verificado se a última movimentação ocorreu a mais do que $Tgossip$ + ($Cfail_j$ * $Tgossip$) unidades de tempo. Caso não, o período de movimentação previamente conhecido ainda não expirou, indicando uma possível movimentação em curso, logo nada é feito e aguarda-se a próxima verificação

de estado. Caso sim, o nó é considerado suspeito e retirado da lista de mobilidade. A espera por um número de intervalos múltiplo de $Cfail_j$ procura computar o tempo de movimentação antes de adicionar o nó como suspeito. Isto faz com que um nó que possui um padrão de mobilidade não seja suspeito por apenas sair do alcance de transmissão.

4.4 Propriedades e Prova de Correção do Algoritmo

O algoritmo de detecção de defeitos com suporte à mobilidade apresentado na Seção 4.3 atende as propriedades de completude forte e exatidão fraca após um tempo definidas em [Chandra e Toueg 1996], ou seja, é um algoritmo da classe ◊S. Tais propriedades asseguram que, após um tempo, os nós defeituosos serão suspeitos pelos nós corretos (completude forte) e que existirá também um instante de tempo *t* após o qual um nó correto não será considerado suspeito por nenhum outro nó (exatidão fraca após um tempo).

A seguir, apresenta-se um argumento de prova para o algoritmo proposto, considerando as propriedades de um detector de defeitos da classe ◊S.

Considere um nó correto p_i e um nó falho p_f . Para que o algoritmo possua a propriedade de *completude forte*, deve-se demonstrar que eventualmente um nó $p_f \in \prod$ será incluído na lista de suspeitos SL_i de todo nó $p_i \in \prod$, sendo $i \neq f$. Esta condição segue das seguintes verificações:

(1) Todo nó p_f pertencente a $\prod$ está inicialmente em SL_i. Para fazer parte de NBL_i o nó p_f deve ter enviado pelo menos uma mensagem *gossip* em um intervalo *Tgossip*.

(2) Um nó p_f não enviará mais mensagens *gossip.* Todo nó p_i espera $Cfail_f$ intervalos antes de adicionar p_f como falho. Logo, existirá um instante de tempo após o qual p_f será adicionado como suspeito em SL_i de todo nó $p_i \in \prod$.

De maneira similar, para demonstração da propriedade de *exatidão fraca após um tempo*, deve-se verificar que existe um instante de tempo *t* a partir do qual um nó correto p_i não vai ser suspeito por nenhum outro nó correto do sistema, ou seja, após o instante *t* o nó p_i não vai fazer parte de nenhum conjunto SL_j. Esta condição é verificada como segue.

(1) Se p_i pertence ao GC_j, após um intervalo *Tgossip*, p_i enviará uma mensagem *gossip* para p_j, que adicionará o nó p_i em sua lista de vizinhos NBL_j. Essa informação será difundida para todos vizinhos p_k do nó p_j, onde $p_k \in GC_j$. Deste modo, todo nó p_i que se movimenta numa trajetória desconhecida, mas que obedece a um padrão de movimentação no qual após um tempo sempre passa na vizinhança de algum nó correto, terá sua informação de sobrevivência difundida para todos os nós corretos. Todo nó correto irá receber informações sobre o nó p_i e remover p_i da lista de suspeitos, adicionando p_i em sua lista de vizinhos NBL_k.

(2) Se o nó p_i estiver na lista de suspeitos de algum nó p_j pertencente a $\prod$, o nó p_i terá a sua informação de $Cfail_i$ atualizada, fazendo com que o nó p_i possa ter o mesmo intervalo de movimentação dentro de sua trajetória sem ser adicionado novamente como suspeito.

5. Testes e Avaliações, Simulações e Comparações

Nesta seção é apresentado o resultado das simulações feitas com o detector de defeitos baseado na potência do sinal (DD). Nos experimentos o modelo de simulação foi configurado para operar num campo de 300m^2, com uma frequência de transmissão de 2.412 ~ 2.462 GHz (IEEE 802.11g) e uma largura de banda de 54 Mbps. O alcance de transmissão de cada nó foi fixado em 50 metros. Para fins de comparação, os parâmetros de simulação foram os mesmos utilizados por Friedman e Tcharny (2005): número de nós, velocidade de movimentação, atraso de comunicação e modelo de movimentação (*Random Waypoint*). Nas simulações foram utilizados diferentes valores para as velocidades máximas de movimentação (de 2 a 8 m/s), número de nós (de 30 a 60 nós).

O primeiro experimento visou identificar o comportamento do número de falsas suspeitas e do tempo de detecção (T_D) frente à variação da quantidade de nós e da velocidade de movimentação. A Figura 4(a) ilustra que o aumento da quantidade de nós e da velocidade dos nós gera um aumento proporcional no número de falsas suspeitas. A proporcionalidade deriva da propriedade linear do algoritmo na manipulação das regiões. A Figura 4(b) ilustra que o T_D decresce rapidamente com o aumento da velocidade de movimentação. Isto deve-se ao fato de que quando se aumenta a velocidade de movimentação dos nós, o parâmetro *CFail* do algoritmos (*timeout*) tende a diminuir, pois se um nó faz uma dada rota, esta será percorrida num menor tempo, diminuindo assim o *CFail*, pois o detector manipula o contador *CFail* com intervalos de espera até classificar um nó como falho.

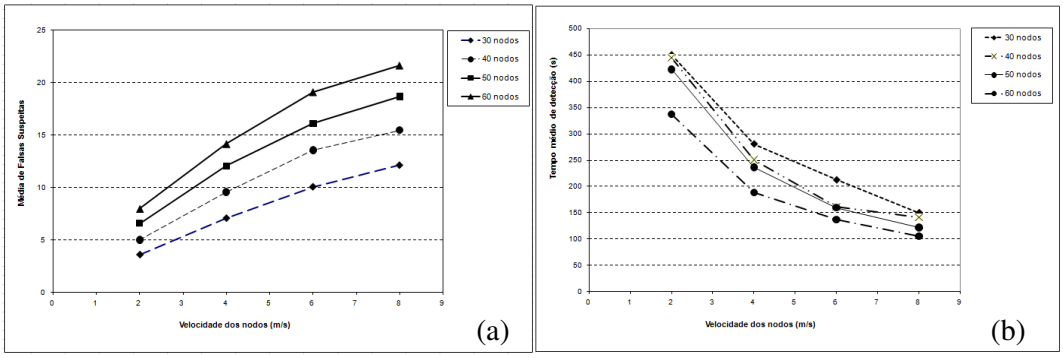


Figura 4. Comportamento do (a) número de falsas suspeitas e (b) tempo médio de detecção

O segundo experimento analisa o comportamento do T_D e da variabilidade do número de falsas suspeitas (Figura 5(a)) e do T_D mínimo frente a variabilidade da velocidade dos nós (Figura 5(b)). Da Figura 5(a), observa-se que o decremento do T_D é proporcional ao aumento do número de falsas suspeitas, o que era esperado. Isso ocorre porque o nó varre sua lista local antes de cada emissão de *broadcast* para enviar sempre sua lista mais atual e, deste modo, tende a suspeitar mais vezes dos nós vizinhos que não enviaram mensagens no último intervalo *CFail*. Quando há menos nós na rede, ocorre menos troca de mensagens (todos nós pertencentes a um dado CG observam as mensagens de *broadcast*), o que tende a aumentar *CFail*. A Figura 5(b) ilustra um tempo de detecção mínimo baixo e relativamente independente do aumento da velocidade dos nós. Isto deve-se ao fato de que quando uma falha ocorre dentro do alcance de transmissão a detecção é rápida, se comparado com o tempo de detecção

quando um nó está fora do alcance de transmissão. É importante observar que este comportamento corresponde a um monitoramento de um nó que se encontra dentro do alcance de transmissão em mais de uma rodada, ou seja, similar ao de uma rede fixa.

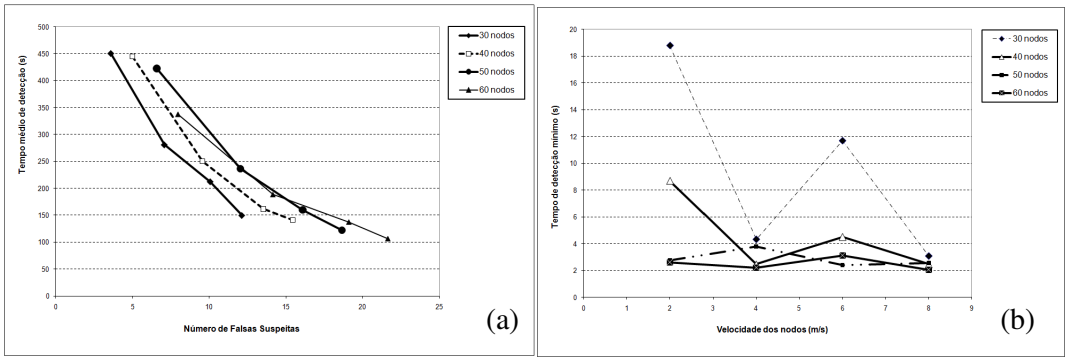


Figura 5. Comportamento (a) do T_D frente a variabilidade do número de falsas suspeitas e (b) do T_D mínimo frente a variabilidade da velocidade dos nós

O terceiro experimento realizou a comparação com outros detectores de defeitos, a fim de identificar o comportamento do número de falsas suspeitas e do tempo de detecção (T_D), confrontando os resultados obtidos pelo detector DD contra os resultados gerados pelos detectores propostos em [Friedman e Tcharny 2005][Sridhar 2006][Sens *et al*. 2008]. Os testes realizados utilizaram uma configuração de rede com 30 nós movendo-se a uma velocidade de 2 m/s. O gráfico da Figura 6(a) demonstra que o DD gerou menos falsas suspeitas em comparação aos outros detectores de defeitos, ficando com um valor muito próximo aos detectores propostos por Sridhar e Friedman. O detector proposto por Pierre Sens corrige este elevado número de falsas suspeitas após suas mensagens de *Query* serem disseminadas na rede, porém o detector comete um elevado número de falsas suspeitas se considerado o tempo total de simulação. O baixo número de falsas suspeitas do DD deve-se ao uso do histórico de movimentação e à utilização do contador *CFail*. No que se refere ao tempo de detecção T_D, a Figura 6(b) ilustra que o detector de Pierre Sens é o que resulta no melhor desempenho de detecção e que o DD consegue resultado melhor que os detectores de Friedman e Sridhar. Este resultado demonstra que o DD é competitivo em relação ao desempenho. Juntos, os resultados em relação ao tempo para detecção de falhas (T_D) e a quantidade de falsas suspeitas, permitem observar que o DD apresenta uma boa qualidade de serviço em relação aos outros detectores similares.

A qualidade do detector de defeitos baseado na potência do sinal (DD) deriva da eliminação de suspeita imediata de um nó p_j que sai do alcance de transmissão de um nó p_i para percorrer uma dada rota de movimentação. O nó percebe uma movimentação de um vizinho que está saindo do seu alcance de transmissão e ajusta o parâmetro *CFail* para aguardar um intervalo de tempo adicional antes de colocar o nó na lista de suspeitos.

Salienta-se que a percepção de movimentação foi realizada a partir de um histórico de regiões que considera as movimentações nos últimos dois intervalos de tempo de transmissão. O intervalo pode ser suficiente caso o nó repita a sua rota num intervalo menor do que duas vezes o período de atualização do histórico, mas deve ser

revisto e é adaptável conforme as novas rotas descobertas (que levam mais tempo). Para adaptar os intervalos de tempo, o DD não precisa conhecer quantos *hops* um nó está do outro, ou informações sobre o *jitter* da rede. Ao invés disto o algoritmo consegue adaptar e obter essas informações apenas observando o funcionamento da rede através da troca de mensagens.

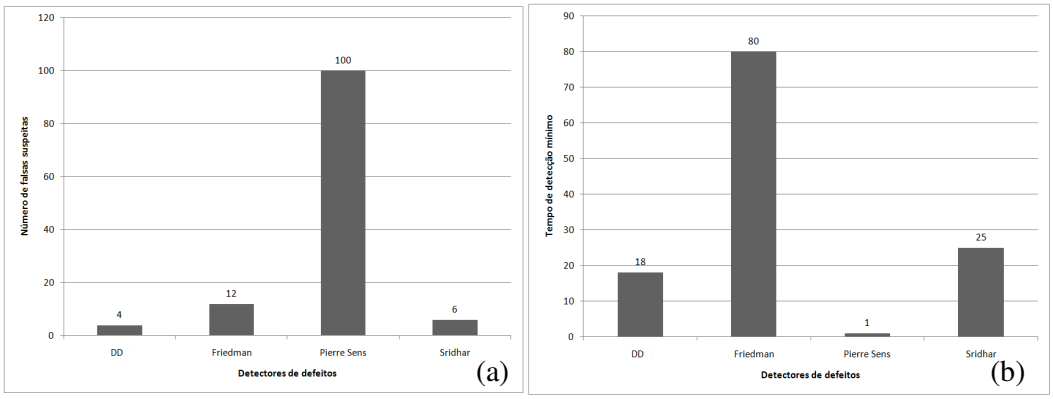


Figura 6. Qualidade do DD frente a outros detectores (a) quanto ao número de falsas suspeitas e (b) quanto ao tempo de detecção (T_D mínimo)

6. Trabalhos Relacionados

Na literatura de detecção de defeitos a mobilidade dos nós foi tratada por diversos autores. Friedman e Tcharny (2005) procuraram representar a mobilidade permitindo que o *Tcleanup* do algoritmo *Gossip* [Renesse *et al.* 1998] pudesse ser incrementado com múltiplos do atraso de rede a um *hop*, suportando assim lacunas no recebimento de mensagens de *heartbeat*. Hutle (2004) também propôs um algoritmo derivado do *Gossip* mas a intenção foi garantir QoS através da inserção de uma variação conhecida do atraso de comunicação (sincronismo fraco). Zia *et al.* (2009) procuraram detectar mobilidade no detector de defeitos usando um *round* adicional de difusão de mensagens e permitindo que os nós que se moveram para fora do alcance de transmissão sejam marcados como suspeitos até que alguma contra-resposta seja obtida. Porém nestas técnicas, a taxa média de erros pode ser alta se os nós se movem frequentemente para fora da região de cobertura.

Alternativamente, Sridhar (2006) propôs um detector de defeitos que trata da mobilidade dos nós mantendo uma lista de nós suspeitos de falha e difundindo essa lista para descobrir se nós marcados como suspeitos não estão em uma outra vizinhança. Além da lista, são usadas duas camadas: uma camada de detecção local (responsável por gerar a lista de vizinhos) e uma camada de detecção de mobilidade. Sens *et al.* (2008) não utilizam fofoca e exploraram a técnica de detecção de defeitos *query-response* para chegar a um detector de defeitos que não depende do monitoramento de tempo.

Diferente de tudo que se encontrou na literatura, este trabalho explorou informações de potência do sinal de recepção que permitem identificar uma

movimentação dos nós vizinhos que estão dentro do alcance de transmissão, habilitando a distinção entre nó em mobilidade e nó defeituoso.

7. Conclusões e Trabalhos Futuros

Detecção de defeitos ainda tem sido alvo de pesquisas para sua utilização eficiente em redes móveis, onde o dinamismo da rede, a escassez de recursos, a comunicação sem fio por difusão de mensagens e a dificuldade de se diferenciar entre um nó defeituoso e um nó móvel são fatores importantes. Este trabalho apresentou uma nova abordagem para implementação de detectores de defeitos não confiável com suporte à mobilidade de nós de redes dinâmicas: distinguir um nó defeituoso de uma mobilidade através da análise da potência do sinal de recepção das mensagens que chegam a um dado nó.

O algoritmo proposto demonstrou ser eficaz quanto a velocidade de detecção de defeitos, quando estes ocorrem em instantes em que o nó defeituoso está dentro do alcance de transmissão do nó monitor. Esta situação é equivalente a uma situação sem mobilidade. Quando a falha ocorre fora do alcance de transmissão de um determinado nó, o tempo máximo de detecção da falha do nó será proporcional ao período de tempo de comunicação (*TGossip*) e será determinado pelo fator *CFail*. Este comportamento é similar aos demais algoritmos que adaptam seu *Timeout*. Por outro lado, em situações de mobilidade que a rota de movimentação segue um padrão de movimentação conhecido (repetição de rota), o algoritmo adapta-se as rotas de cada nó rapidamente e evita falsas suspeitas quando um nó correto sai do alcance de transmissão apenas para fazer a sua rota. Neste cenário o resultado observado foi um baixo número de falsas detecções quando comparado com algoritmos similares.

Em trabalhos futuros vislumbra-se explorar: (1) a identificação de mobilidade no próprio nó, o que pode permitir que ele informe outros nós sobre sua movimentação ou mesmo permitir a trocar de comportamento do detector; (2) o impacto do refinamento do tamanho das regiões consideradas e do tamanho do histórico, visando instrumentalizar o detector com uma noção mais precisa da rota de movimentação.

Referências

Aguilera, M. K.; Chen, W.; Toueg, S. (1997) “Heartbeat: A timeout-free failure detector for quiescent reliable communication”. Workshop on Distributed Algorithms, pp. 126–140.

Aguilera, M. K.; Chen, W.; Toueg, S. (1998) “Failure Detection and Consensus in the Crash-Recovery Model”. Technical Report 98-1676. Department of Computer Science, Cornell University, v. 1499, pp.231-245.

Burns, M.; George, A; Wallace, B. (1999) “Simulative Performance Analysis of Gossip Failure Detection for Scalable Distributed Systems”. *Cluster Computing*. 2(3):207-217.

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

Chen, W; Toueg, S.; Aguilera, M. K. (2002) “On the quality of service of failure detectors”, *IEEE Transactions on Computing*, 51(1):13-32.

Felber, P.; Défago, X.; Guerraoui, R.; Oser, P. (1999) “Failure detectors as first class objects”. In: International Symposium on Distributed Objects and Applications (DOA’99). Edinburgh, Scotland. pp.132–141. IEEE Computer Society.

Friedman, R.; Gavidia, D.; Rodrigues, L.; Viana, A. C.; Voulgaris, S. (2007) “Gossiping on MANETs: the Beauty and the Beast”, *ACM Operating Systems Review*, pp.67-74.

Friedman, R.; Tcharny, G. (2005), "Evaluating failure detection in mobile ad-hoc networks", *International Journal of Wireless and Mobile Computing*, 1(8):23.

Fischer, M. J.; Lynch, N. A.; Paterson, M. S. (1985) “Impossibility of distributed consensus with one faulty process”. *Journal of the ACM*, v.32, n.2, pp. 374-382.

Gomez, J.; Campbell, A. T. (2007) “Using Variable-Range Transmission Power Control in Wireless Ad Hoc Networks”, *IEEE Trans on Mobile Computing*, 6(1), pp.87-99.

Gracioli, G.; Nunes, R. C. (2007) “Detecção de defeitos em redes móveis sem fio: uma avaliação entre as estratégias e seus algoritmos”. In: Anais do Workshop de Testes e Tolerância a Falhas (SBRC/WTF), Belém/PA, pp.159-172.

Hutle, M. (2004) “An efficient failure detector for sparsely connected networks”. In: Proceedings of the International Conference on Parallel and Distributed Computing and Networks. Innsbruck: Austria, pp.369-374.

Loo, J.; Mauri, J. L.; Ortiz, J. H. (2012) “Mobile Ad Hoc Networks: current status and futures trends”. CRC Press, 538p.

Mahapatro, A.; Khilar, P.M., (2013) “Fault Diagnosis in Wireless Sensor Networks: A Survey,” *Communications Surveys & Tutorials*, IEEE , vol.15, no.4, pp.2000-2026.

Renesse, R.; Minsky, Y.; Hayden, M. (1998) “A Gossip-style failure detection service”. In: Proceedings of the IFIP Int. Conf. on Distributed Systems and Platforms and Open Distributed Processing (Middleware’ 98), pp.55-70.

Sens, P., Greve, F., Arantes, L., Bouillaguet, M., Simon. V. (2008) “Um detector de falhas assíncrono para redes móveis e auto-organizáveis”. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC’ 2008, pp.931-944.

Sridhar, N. (2006) “Decentralized local failure detection in dynamic distributed systems”. In: IEEE Symp on Reliable Distributed Systems (SRDS'06). pp.143-154.

Subramaniyan, R.; Raman, P.; George, A. D.; Radlinski, M. (2006) “GEMS: Gossip-Enabled Monitoring Service for Scalable Heterogeneous Distributed Systems”. *Cluster Computing*, 9(1), pp.101-120.

Zia, H. A.; Sridhar, N; Sastry, S. (2009) “Failure detectors for wireless sensor-actuator systems”. *Ad Hoc Networks*, 7:1001-1013.