Detecção de Defeitos em Redes Móveis Sem Fio: Uma Avaliação entre as Estratégias e seus Algoritmos*

Giovani Gracioli e Raul Ceretta Nunes

[1]Laboratório de Tolerância a Falhas
GMICRO/CT – Universidade Federal de Santa Maria (UFSM)
Campus Camobi - 97105-900 – Santa Maria/RS

{giovani,ceretta}@inf.ufsm.br

Resumo. *Recentemente, o avanço no uso de tecnologias de redes sem fio tem proliferado o número de redes móveis sem fio que operam em modo* ad hoc. *Por caracterizar-se como uma classe particular de sistemas distribuídos, onde os atrasos e perdas de mensagens são mais significativos e onde há mobilidade de nodos, redes móveis sem fio demandam algoritmos especializados. Este artigo avalia as características dos principais algoritmos de detecção de defeitos para redes móveis sem fio e os compara em relação ao número de* broadcasts *gerados e em relação a qualidade de serviço apresentada. Como resultado observa-se que a escolha do algoritmo deve ser guiada pela necessidade da aplicação combinada com a capacidade de energia da rede.*

Abstract. *Recently, the advance in the use of wireless technologies increased the number of wireless networks that operate in ad hoc mode. As a particular class of distributed systems, where delays and message loss are more significant and where there is nodes mobility, wireless ad hoc networks demand specialized algorithms. This paper evaluates the characteristics of failure detector algorithms for wireless ad hoc networks and compare them from number of generated broadcasts and quality of service. As result we show that the algorithm choice must be driven by application requirement and network energy capability.*

1. Introdução

O crescimento tecnológico ocorrido nos últimos anos na área de redes sem fio criou demanda pela melhoria na qualidade dos *softwares* e *hardwares* para redes sem fio [Mateus and Loureiro 2005]. A necessidade de trocar informações de maneira simples e direta, faz com que as redes *ad hoc* sem fio, também conhecidas como MANETs (*Mobile Ad hoc NETworks*), sejam cada vez mais utilizadas quando trata-se de celulares, PDAs ou notebooks. Redes *ad hoc* não exigem infra-estrutura para a comunicação, pois seus nodos podem comunicar-se diretamente enquanto o alcance de transmissão de seu rádio permitir comunicação com vizinho(s). Do mesmo modo, MANETs formadas por nodos sensores (nodo com sensor, processador e rádio) encontram uma diversidade de aplicações como, por exemplo, em monitoramento ambiental. Uma Rede de Sensores Sem Fio (RSSF) caracteriza-se por sua mobilidade, limitações no consumo de energia e grande número de nodos [Pereira et al. 2003] e pode ser definida como uma classe particular de sistemas

*Projeto apoiado pela Fundação de Amparo a Pesquisa do RS - FAPERGS (Proc. 05/51231.1).

distribuídos [Heidemann et al. 2001]. Neste artigo referencia-se uma MANET ou RSSF simplesmente por Rede Móvel Sem Fio (RMSF).

Por definição, RMSFs possuem a mesma redundância intrínseca encontrada em sistemas distribuídos, o que é desejável para sistemas tolerantes a falhas, mas a mobilidade e a maior vulnerabilidade a perda de mensagens dificulta a reutilização direta de algoritmos e protocolos já desenvolvidos para sistemas distribuídos que executam em redes tradicionais. Como conseqüência, os algoritmos fundamentais dos mecanismo de tolerância a falhas, também chamados de blocos básicos de construção [Jalote 1994], necessitam ser reavaliados.

O detector de defeitos, que é um algoritmo distribuído que fornece informações sobre suspeitas de defeitos em componentes monitorados [Felber et al. 1999], é um bloco básico importante na construção de protocolos de acordo [Chandra and Toueg 1996], fundamentais na implementação de técnicas de tolerância a falhas. Duas propriedades caracterizam um detector de defeitos [Chandra and Toueg 1996]: a abrangência (*completeness*) e a exatidão (*accuracy*). A abrangência se refere a capacidade para detectar defeitos de nodos que realmente falharam, enquanto que a exatidão se refere a capacidade de não cometer falsas suspeitas. A especificidade de cada propriedade num dado algoritmo de detecção determina sua classe de detecção.

Diversos algoritmos para detecção de defeitos em RMSF foram propostos recentemente [Tai et al. 2004, Hutle 2004, Friedman and Tcharny 2005] e foram alvo de trabalho prévio [Gracioli and Nunes 2006], onde observa-se que considerando o modelo de comunicação adotado, os algoritmos podem ser agrupados em duas estratégias: baseados em *cluster* [Tai et al. 2004] e baseados em fofocas (*gossip*) [Hutle 2004, Friedman and Tcharny 2005]. Os baseados em fofocas, por derivarem do algoritmo Gossip [Renesse et al. 1998], são referidos neste texto como *baseados no Gossip*. Porém, uma avaliação qualitativa entre as estratégias e seus algoritmos ainda não é conhecida. Este artigo avalia as duas estratégias e seus algoritmos através da comparação dos detector de defeitos, incluindo o Gossip, considerando as métricas primárias de qualidade de serviço para detectores de defeitos [Chen et al. 2002] e o número de *broadcasts* realizado por cada algoritmo.

Este artigo está organizado da seguinte maneira. Na seção 2, as duas estratégias e os algoritmos que fazem parte delas são apresentados em maiores detalhes. Na seção 3, são apresentadas as avaliações e os testes realizados. As discussões dos principais resultados, bem como dos trabalhos futuros, são apresentados na seção 5.

2. Estratégias para Detecção de Defeitos em RMSF

De acordo com o modelo de comunicação, detectores de defeitos para RMSF podem ser classificados em duas estratégias [Gracioli and Nunes 2006]: baseados em *cluster* e baseados no algoritmo *Gossip*. Nesta seção cada estratégia é apresentada através de seus algoritmos.

2.1. Detector de Defeitos Baseado em Cluster

Num detector de defeitos que usa um modelo de comunicação baseado em *cluster*, um *cluster* pode ser visto como um círculo com o raio igual ao alcance de transmissão do nodo central, chamado de *clusterhead* (CH). Cada *cluster* tem somente um CH e a função do

CH dentro do seu *cluster* é detectar os nodos defeituosos. Como numa RMSF pode existir vários *clusters* um nodo que consegue se comunicar com dois *clusterheads* é chamado de *gateway* (GW) e é utilizado para transmitir informações de estado entre *cluster*. Os nodos que não são nem CHs e nem GWs são chamados de membros ordinários (OM). Para a escolha dos CHs, GWs e OMs um algoritmo de formação de *cluster* (AFC) precisa ser utilizado. Os principais objetivos desse algoritmo são: permitir que a comunicação seja robusta, escalável e eficiente; e resolver os problemas da mobilidade e escalabilidade dos nodos que são comuns em RMSF [Tai and Tso 2004].

No detector de defeitos baseado em *cluster* proposto por [Tai et al. 2004] destaca-se dois algoritmos: um responsável pela formação do *cluster* e outro responsável pelo serviço de detecção de defeitos (SDD).

O AFC é uma variante do algoritmo *Lowest-ID* [Gerla and Tsai 1995] e possui as seguintes propriedades:

- (*i*) assegura que um GW é afiliado a um e somente um *cluster* e que existirão vários nodos candidatos a GW;
- (*ii*) cria CHs substitutos (*Deputy Clusterheads* - DCHs) e *backup gateways* (BGWs) que deixam o detector mais flexível a defeitos dos nodos;
- e (*iii*) permite que as informações coletadas na primeira rodada do AFC sejam utilizadas também na primeira rodada do algoritmo do SDD[1].

O algoritmo do SDD é dividido em duas etapas: *intra-cluster* e *inter-cluster*. Na etapa *intra-cluster*, ele é baseado em *heartbeats* e consiste em 3 fases. Na primeira, cada membro de um *cluster* local *C* envia uma mensagem *heartbeat* contendo o seu NID (identificador do nodo). A mensagem corresponde a um *broadcast* que alcança o CH e os nodos vizinhos (dentro do alcance de transmissão do nodo). Na segunda, todo nodo envia para o seu CH, também via *broadcast*, um relatório de todos os *heartbeats* dos nodos que ele conseguiu ouvir durante a primeira fase. Na terceira, analisando as informações coletadas na primeira e segunda fases, o CH identifica os nodos suspeitos e então transmite uma mensagem de atualização para o *cluster*, indicando sua decisão. Um nodo *v* é determinado falho se e somente se o CH não receber nem o *heartbeat* de *v* na primeira fase, nem o relatório de *v* na segunda fase e nenhum dos relatórios recebidos pelo CH na segunda fase refletir o conhecimento do *heartbeat* pertencente ao *v*. Um CH, por sua vez, é julgado suspeito se e somente se o DCH não receber nem o *heartbeat* do CH na primeira fase, nem o relatório do CH na segunda fase, nenhum dos relatórios recebidos pelo DCH conter informações sobre o *heartbeat* do CH e se o DCH não receber a mensagem de atualização dos estados enviada pelo CH na terceira fase. Para garantir a propagação da informação entre *clusters* distintos, após a terceira fase, o algoritmo *inter-cluster* do SDD consiste em repassar a outros *clusters*, via GW, a informação sobre suspeitos.

Em síntese, um detector de defeitos baseado em *cluster* precisa ter um algoritmo de formação de *cluster* e um algoritmo para detecção de defeitos *intra-cluster* com disseminação de informações *inter-cluster*.

2.2. Detectores de Defeitos Baseados no Algoritmo Gossip

No algoritmo Gossip [Renesse et al. 1998] cada nodo da rede é um monitor/monitorado e mantém uma lista contendo o endereço e um inteiro (contador de *heartbeat*) para cada

[1]Depois da primeira rodada, os algoritmos de formação de cluster e do SDD executarão separadamente.

nodo do sistema. A cada intervalo T_{gossip}, um nodo escolhe aleatoriamente um ou mais vizinhos para enviar a sua lista. A lista recebida é unida com a lista que o receptor possui e o maior valor do contador de *heartbeat* presente nas listas é mantido na lista final. Cada nodo mantém o instante do último incremento do contador de *heartbeat*. Se o contador não for incrementado em um intervalo de T_{fail} unidades de tempo então o membro é considerado suspeito, mas colocado na lista de suspeitos apenas após $T_{cleanup}$ unidades de tempo. O ponto chave do algoritmo é que ele independe de topologia, pois um nodo conhece seus vizinhos ao longo do tempo através de mensagens que passam de vizinhos para vizinhos [Renesse et al. 1998]. Por esta razão o algoritmo é capaz de suportar a mobilidade dos nodos em redes móveis e por isso serve como base para outros detectores de defeitos propostos para RMSF e que são detalhados a seguir.

Em [Friedman and Tcharny 2005] foi proposta uma adaptação do modelo de detecção de defeitos *Gossip* para redes móveis sem fio. O algoritmo usa um contador de *heartbeat* que é incrementado toda vez que um novo *heartbeat* de algum vizinho é recebido. Na teoria, a cada π unidades de tempo (equivalente a um *timeout*) um *heartbeat* deveria ser recebido, mas na prática isso pode não acontecer devido ao problema da mobilidade ou devido ao aumento do caminho (maior quantidade de nodos presente na rede). Para evitar falsas suspeitas, o algoritmo então aguarda por um máximo de γ *heartbeats* antes de suspeitar de um nodo entre o recebimento de dois *heartbeats* consecutivos. O algoritmo espera que os nodos estejam em movimento, hora fazendo parte da rede, hora estando fora do alcance de transmissão. O principal desafio é encontrar um valor de γ tal que com o passar das rodadas mesmo que um nodo vá para fora do alcance de transmissão, este nodo não seja suspeito, pois o valor de γ deveria estar ajustado para o tempo desse movimento. O algoritmo mantém as seguintes propriedades:

- garante que todos os processos defeituosos serão eventualmente suspeitos;
- a cada π unidades de tempo o número total de mensagens enviadas é *O(n)*, sendo *n* o número de nodos da rede;
- o algoritmo tem um melhor desempenho quando a conectividade e o número de nodos da rede aumentam. Isso se deve ao fato de que quando a conectividade da rede é ruim, a disseminação de uma detecção de defeito realizada por um nodo alcançará um pequeno número de vizinhos criando um número maior de falsas detecções. Se o número de nodos da rede aumenta, conseqüentemente a conectividade aumenta e a característica epidêmica do protocolo melhora o desempenho.

Outro algoritmo baseado no *Gossip* foi proposto em [Hutle 2004]. O algoritmo foi proposto para redes esparsamente conectadas, onde os nodos podem somente se comunicar com seus vizinhos via *broadcast* sem fio, e também baseia-se em rodadas locais e repasse de informações (fofocas). No algoritmo, o número total de nodos do sistema não precisa ser conhecido, mas um nodo necessita conhecer o *jitter* (variação de atraso) na comunicação entre seus vizinhos diretos. Esse conhecimento na variação de atraso necessita um sincronismo na comunicação direta entre dois nodos vizinhos, o que pode ser obtido reservando um limite na largura de banda para o detector de defeitos e limitando o seu número de mensagens. No algoritmo, cada nodo *p* mantém para todo outro nodo *q* um contador de *heartbeats*, um contador de distância (que contém a estimativa sobre a atual distância de *q*) e um vetor de *time-stamp* que mantém a última rodada que *p* recebeu um *heartbeat* de *q*. Através do contador de distância, o algoritmo obtém uma noção de

quantos *hops* de distância um nodo está de outro e conseqüentemente quantos repasses precisa realizar.

2.3. Discussão das Estratégias

Um detector de defeitos com uma arquitetura de comunicação baseada em *cluster* permite que haja uma visão sobre a hierarquia dos nodos na rede (CH, GW e OM). Permite também, que um algoritmo de detecção de defeitos seja executado paralelamente dentro do *cluster* [Tai et al. 2004]. Em cada *cluster* existe uma divisão das tarefas, o AFC é responsável por tratar do problema da mobilidade e escalabilidade dos nodos enquanto o algoritmo do SDD é responsável em detectar o defeito. O resultado de uma detecção dentro de um *cluster* é repassado para outros *clusters* através dos GWs, de maneira que o detector explora a redundância de mensagens, que é comum em redes móveis sem fio, o que atenua os efeitos do problema da vulnerabilidade a perda de mensagens e deixa o SDD mais robusto.

Por outro lado, os detectores de defeitos baseados no protocolo *gossip* são mais flexíveis, pois não dependem de uma hierarquia ou topologia fixa, tal como a de *clusters*. Desta forma dispensam a necessidade de algoritmos de formação de hierarquias (AFC), pois a configuração dos nodos é feita dinamicamente através da troca de mensagens, tratando do problema da mobilidade. Além disto, cada nodo difunde as suas informações para um ou mais de seus vizinhos fazendo com que essas informações possam chegar para um outro nodo que não faz parte da sua vizinhança, tornando a detecção abrangente. As vezes, tendem a sacrificar o alcance do nodo pela velocidade ou eficiência [Tai and Tso 2004]. Em geral são mais simples de serem implementados.

3. Avaliações e Testes

Para avaliar as estratégias apresentadas na seção 2, quatro métricas foram utilizadas: as três métricas primárias para avaliar a qualidade de serviço dos detectores [Chen et al. 2002] e uma relacionada o consumo de energia.

- **Tempo de Recorrência ao Erro** (*Mistake recurrence time* T_{MR}): mede o tempo entre dois erros consecutivos cometidos pelo detector (suspeita incorreta).
- **Tempo Médio de Duração de Falsas Suspeitas** (*Mistake Duration* T_M): mede o tempo que o detector de defeitos leva para corrigir um erro.
- **Tempo de Detecção** (*Detection Time* T_D): mede o tempo para que todos os nodos suspeitem de um nodo defeituoso.
- **Número de *Broadcasts***: mede o número de *broadcasts* em cada algoritmo para poder ter uma estimativa do gasto de energia através do uso do rádio.

O ambiente de testes foi o simulador *JiST/SWANS*, na sua versão 1.0.6, suportado pela JVM *Java* 1.4.2_08. No simulador, o modelo de mobilidade dos nodos utilizado foi o *Random Waypoint* [Johnson and Maltz 1996], onde o nodo escolhe uma direção aleatória e uma velocidade entre os valores mínimo e máximo disponíveis e se move até o ponto de destino dentro do campo de simulação. Ao chegar no destino, o nodo permanece parado por um tempo definido (tempo de pausa) e após o término desse tempo repete o processo novamente. Quanto maior é o tempo de pausa mais estática é a rede e quanto mais este valor se aproxima a zero maior é a mobilidade existente na rede.

Assumindo um modelo de falhas do tipo colapso, na simulação falhas são simuladas interrompendo definitivamente o envio de mensagens. A mobilidade foi configurada com um tempo de pausa 15 segundos,[2] com velocidade variando de 0 m/s (nodo parado) até 2 m/s. Todos os testes foram realizados em um campo de 300m x 300m em uma simulação de 30 minutos e variando o número de nodos e o alcance de transmissão.

Entretanto, como os algoritmos diferem em alguns parâmetros de configuração, foi realizado inicialmente testes para a escolher os parâmetros dos algoritmos (subseção 3.1), para então realizar a comparação (subseção 3.2).

3.1. Escolha dos Parâmetros dos Algoritmos

O objetivo dessa etapa é verificar qual são os melhores parâmetros para cada algoritmo, considerando as métricas utilizadas. No caso do algoritmo *Gossip*, procura-se o T_{gossip} e o $T_{cleanup}$[3]. Observa-se que os valores encontrados também servem para configurar os outros dois algoritmos baseados no *Gossip*. No algoritmo baseado em *cluster*, busca-se obter o melhor valor para o tempo (*timeout*) entre a primeira e segunda fases (T_{fases}) e também o tempo entre duas primeiras fases consecutivas (T_{SDD}).

3.1.1. Valores para o Algoritmo Gossip

O algoritmo *Gossip* possui parâmetros que são usados para o seu funcionamento. A cada intervalo de tempo T_{gossip} o nodo escolhe aleatoriamente um ou mais vizinhos para enviar a sua lista e a cada $T_{cleanup}$ unidades de tempo o nodo varre a sua lista procurando por vizinhos que não atualizaram seus contadores de *heartbeats*. Porém, para o ambiente de teste apresentado anteriormente, esses valores ainda não são conhecidos. Por esta razão, a primeira tarefa é escolher quais os valores para T_{gossip} e $T_{cleanup}$ que melhor se encaixam no ambiente de simulação. Para isso usou-se 20 nodos com alcance de transmissão de 25 metros e os valores para T_{gossip} foram fixados em 5, 8, 10, 12 e 15 segundos. Já os valores para o $T_{cleanup}$ foram atribuídos como sendo igual, 2 e 3 vezes o valor do T_{gossip}.

O primeiro teste avaliou o tempo médio de recorrência ao erro (T_{MR}), onde pode-se perceber (vide figura 1) que o T_{MR} é diretamente proporcional ao valor do $T_{cleanup}$. Quanto maior o $T_{cleanup}$, maior o T_{MR}. Isso se deve ao fato que um nodo com um valor para o $T_{cleanup}$ igual a 3 vezes o T_{gossip}, varre a sua lista procurando por nodos que não atualizaram seu contador de *heartbetas* menos vezes do que um nodo que tem o valor igual ao T_{gossip}.

A seguir o tempo médio de duração de falsas suspeitas foi medido e observa-se que com o aumento do T_{gossip}, o T_M também sofre um aumento (vide figura 1). Já para o $T_{cleanup}$, o valor de T_M também sofre uma variação, mas essa variação é menor se comparada com o T_{gossip}. O T_{gossip} tem uma maior influência neste caso pois aumenta o intervalo entre o envio da lista de vizinhos de cada nodo, com isso, aumenta também o tempo que um nodo leva para obter ciência que sua suspeita está incorreta.

[2]Embora a variação do tempo de pausa possa ser um fator relevante, ele está diretamente relacionado a aspectos de mobilidade da rede e será alvo de estudos futuros.

[3]Segundo [Subramaniyan et al. 2005] o $T_{cleanup}$ pode substituir o T_{fail} e pode ser definido como múltiplo do T_{gossip}.

A figura 1 mostra também um aumento linear no tempo para detecção de um defeito (T_D) conforme o T_{gossip} e o $T_{cleanup}$ aumentam. O T_D está diretamente ligado nas freqüências de envio das mensagens e na procura por nodos que não incrementaram seus contadores de *heartbeats*. Então com um aumento do T_{gossip} e do $T_{cleanup}$, o T_D também se torna maior.

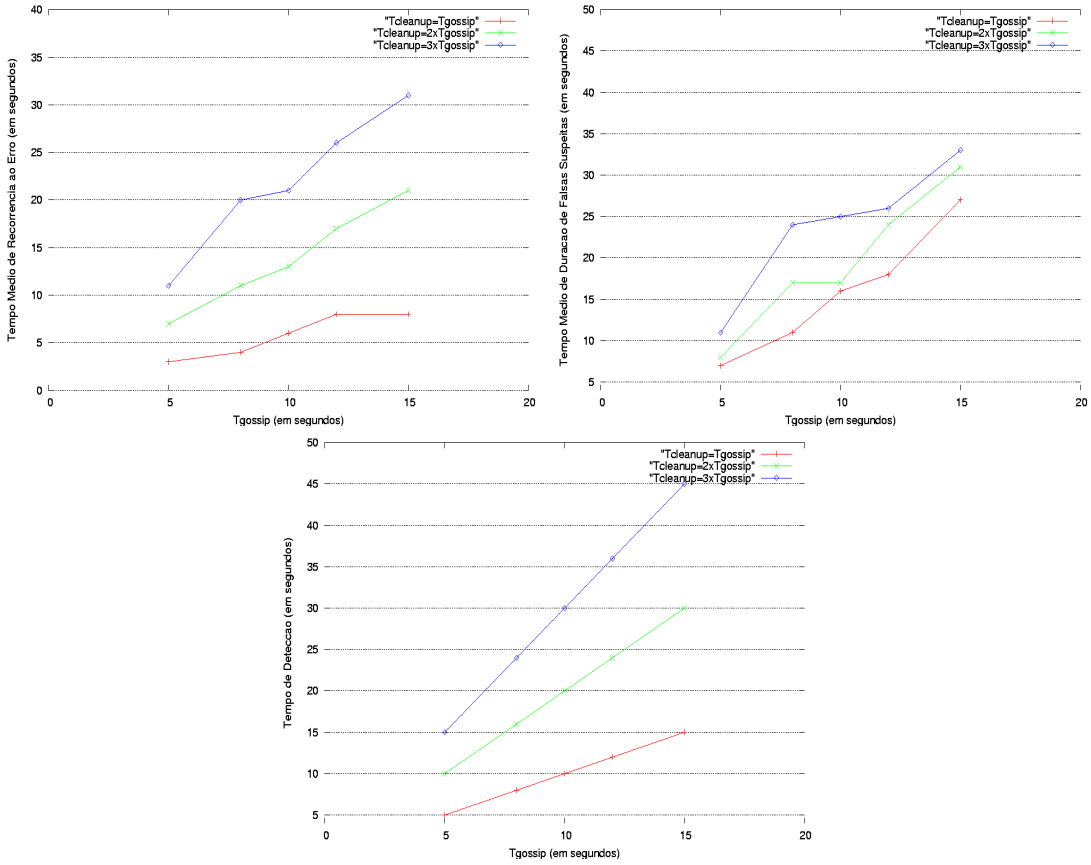

Figura 1. Métricas T_{MR}, T_M e T_D do algoritmo *Gossip*.

Na métrica T_{MR}, busca-se um valor mais alto para diminuir a recorrência ao erro. Já nas métrica de T_D e T_M, valores mais baixo são desejados, com isso o detector de defeitos tomaria ciência do erro mais rapidamente e seria mais confiável, pois um nodo permaneceria suspeito por um menor tempo. Com base nisso, o valor do T_{gossip} escolhido foi 12 segundos e o valor para o $T_{cleanup}$ foi 2 vezes o valor do T_{gossip}, pois são os valores que apresentaram um desempenho mais regular nas 3 métricas.

3.1.2. Valores para o Algoritmo Baseado em Cluster

Neste algoritmo, são realizadas três fases, dentre as quais na primeira, o nodo envia um *broadcast* contendo seu NID e na segunda o nodo envia a lista de todos os vizinhos que foi possível receber o NID na primeira fase. O tempo (*timeout*) entre a primeira e segunda fases é chamado de T_{fases} e o tempo entre duas primeiras fases consecutivas (tempo entre o envio de dois *broadcasts*) é chamado de T_{SDD}. Assim como o algoritmo *Gossip*, esses valores não são conhecidos, portanto, deve-se obter os melhores valores para o T_{fases} e

T_{SDD} para o ambiente de simulação. Em todos os testes realizados neste algoritmo o T_{SDD} tem o valor igual ao T_{gossip}, ou seja, 12 segundos[4]. No entanto, os valores considerados para o T_{fases} são relacionados com o T_{SDD} e são apresentados na tabela 1.

Tabela 1. Valores considerados para o T_{fases}.

Relação entre o T_{fases} e o T_{SDD}	Valor em Segundos
T_{fases} = T_{SDD} / 3	4
T_{fases} = T_{SDD} / 2	6
T_{fases} = T_{SDD} * 3 / 4	9
T_{fases} = T_{SDD}	12

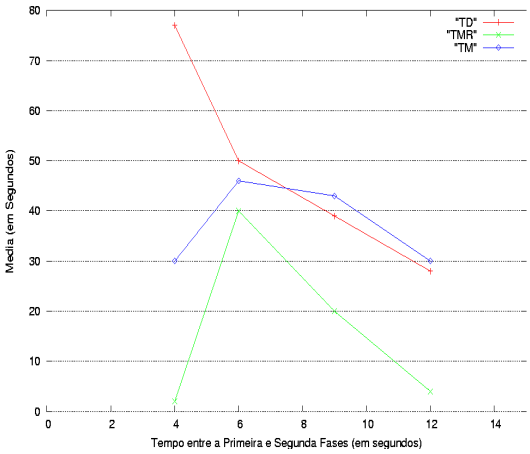

Figura 2. Métricas T_D, T_{MR} e T_M para o algoritmo baseado em *Cluster*.

A figura 2 apresenta as médias em segundos para o tempo de detecção, tempo médio de recorrência ao erro e tempo médio de duração de falsas suspeitas. O T_D mostrou-se inversamente proporcional ao T_{fases}, ou seja, diminuiu o seu valor com o aumento do T_{fases}. Isso se deve ao fato de que quanto mais o T_{fases} se aproxima do T_{SDD} as duas fases vão se tornando uma só e assim o T_D diminui. Já o T_{MR} começou em um valor muito ruim e atingiu o seu melhor estado com um valor para o T_{fases} de 6 segundos. Cabe ressaltar que quanto mais alto o valor para o T_{MR}, melhor é para o detector de defeitos, pois menos erros são cometidos. Analisando o T_M, os melhores valores são alcançados com T_{fases} igual a 4 e 12 segundos. A explicação para isso é que com um T_{fases} igual a 4 segundos mais vezes a mensagem contendo a lista que pôde ser recebida na primeira fase pode ser transmitida na segunda fase. Já para o T_{fases} igual a 12 segundos, as duas fases são unidas em uma só, então, em um mesmo *broadcast*, é possível enviar o NID e a lista recebida.

Em uma análise das métricas do T_D, T_{MR} e T_M o valor para o T_{fases} escolhido foi 6 segundos, pois foi o valor que menos variou nas 3 métricas. Portanto, após definidos os valores de T_{gossip} (12 segundos), $T_{cleanup}$ (2 vezes T_{gossip}), T_{SDD} (12 segundos) e T_{fases} (6 segundos) é possível comparar os algoritmos. Os resultados dessa comparação são apresentados na próxima seção.

[4]A idéia de ter T_{SDD} igual ao T_{gossip} é para manter o tempo de envio entre dois *broadcasts* consecutivos iguais e ver como os algoritmos reagem a esta situação.

3.2. Comparação entre as Estratégias

Nesta seção são comparados os algoritmos pertencentes as duas estratégias de detecção de defeitos para RMSF. Para tal, são consideradas as métricas T_D, T_{MR} e T_M em simulações variando o número de nodos (20, 30, 40, 50 e 60) e o alcance de transmissão (25, 50 e 75 metros), bem como o número de *broadcasts*.

3.2.1. Tempo Médio de Recorrência ao Erro

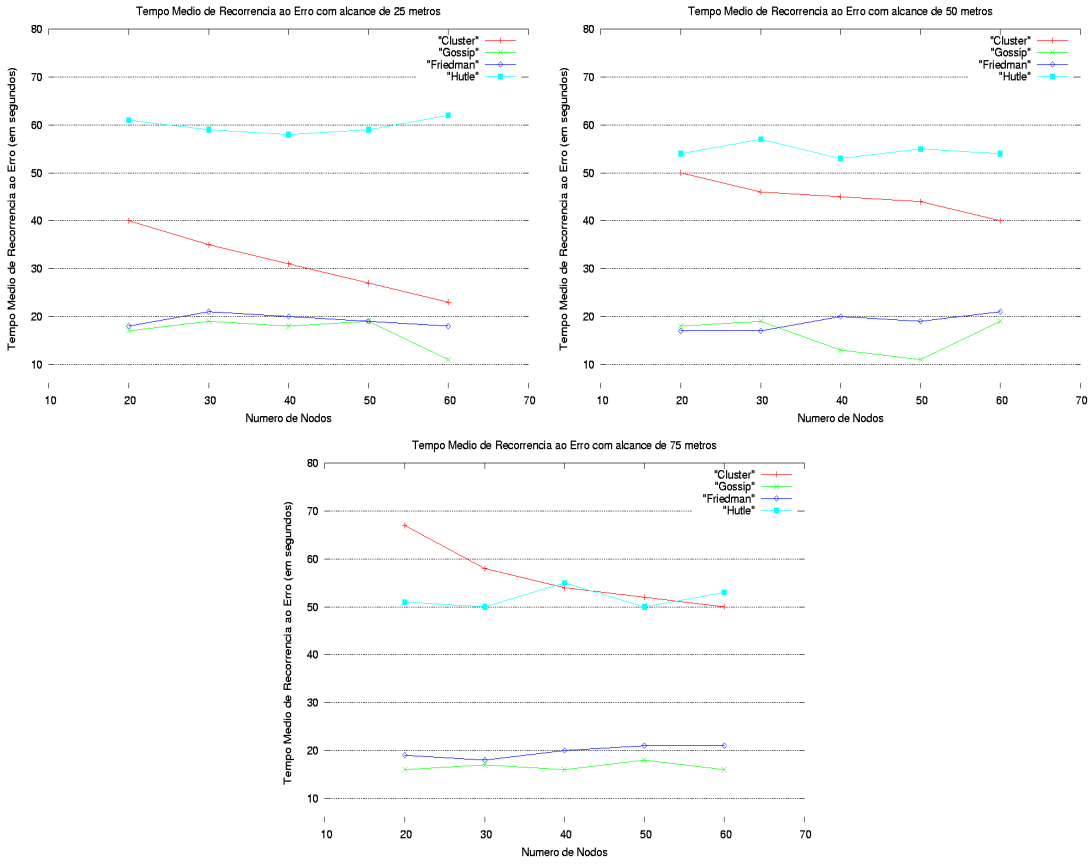

Figura 3. Tempo médio de recorrência ao erro para alcances de transmissão dos nodos de 25, 50 e 75 metros.

Considerando o T_{MR}, quanto maior for o seu valor, mais preciso é o detector de defeitos, pois menos vezes um nodo será considerado suspeito. A figura 3 apresenta os valores do T_{MR} com alcance de transmissão dos nodos de 25, 50 e 75 metros.

Analisando o gráfico para 25 metros, pode-se perceber que o algoritmo proposto pelo *Hutle* tem os melhores valores para esta métrica. Já o detector baseado em *cluster* apresenta um declínio em seus valores conforme o número de nodos aumenta, o que não acontece com os detectores baseado no algoritmo *Gossip* e com o próprio algoritmo *Gossip*, pois seus valores apresentam uma pequena variação para mais ou para menos e são praticamente constantes.

Analisando o gráfico para 50 metros, observa-se que novamente o detector proposto pelo *Hutle* obteve os melhores resultados. Porém, esses valores são um pouco mais

baixos se comparado com os valores do alcance de transmissão de 25 metros, pois com um maior alcance de transmissão os nodos se comunicam com mais vizinhos e assim a probabilidade de considerar suspeitos aumenta. Com o detector baseado em *cluster*, os valores do T_{MR} melhoraram com o aumento do alcance de transmissão. Já os valores para o algoritmo *Gossip* e o de *Friedman* são novamente inferiores aos outros dois detectores e demonstram sofrer pouca influência do alcance de transmissão e aumento do número de nodos.

Os valores do T_{MR} para um alcance de transmissão dos nodos de 75 metros (vide figura 3) demonstram que o detector de defeitos baseado em *cluster* apresenta melhores resultados com 20, 30 ou 50 nodos. Pode ser concluído, que neste detector, conforme o alcance de transmissão aumenta, o T_{MR} também aumenta, porém conforme o número de nodos aumenta o T_{MR} diminui, pois existe uma maior probabilidade de recorrer ao erro. Já com o algoritmo proposto pelo *Hutle*, o T_{MR} se mantém praticamente constante com o aumento no número de nodos, mas diminui com o aumento do alcance de transmissão. Isso pode ser atribuído a maior disseminação da informação que ocorre quando o alcance de transmissão é maior. Por outro lado, o algoritmo *Gossip* e o detector proposto pelo *Friedman* não obteram grande mudança em seu comportamento mesmo com a variação no número de nodos e alcance de transmissão.

3.2.2. Tempo Médio de Duração de Falsas Suspeitas

O tempo médio de duração de falsas suspeitas mede o tempo que o detector de defeitos leva para corrigir um erro, ou seja, é o tempo de duração de um erro. Quanto menor for esse tempo, mais confiável e exato é o detector de defeitos.

Com um alcance de transmissão de 25 metros, observa-se (figura 4) que os melhores resultados para o T_M são do algoritmo proposto pelo *Friedman*. Um comportamento semelhante também foi obtido pelo algoritmo *Gossip*. Já o detector de defeitos baseado em *cluster* obteve um aumento no valor da métrica conforme o acréscimo no número de nodos mais significativo do que os algoritmos baseado no *Gossip* e também apresentou o pior resultado para uma rede com 60 nodos.

Já para 50 metros pode-se perceber (figura 4) que todos os algoritmos obtiveram uma diminuição nos valores do T_M. Isso é explicado devido ao maior alcance e portanto a comunicação ser feita com mais vizinhos. Mais uma vez os algoritmos de *Friedman* e o *Gossip* obtiveram os melhores valores. Por outro lado, o detector baseado em *cluster* obteve um melhor resultado do que o detector proposto pelo *Hutle* mesmo em uma rede com 60 nodos.

Para um alcance de transmissão dos nodos de 75 metros (figura 4), novamente todos os algoritmos diminuíram os seus valores para o T_M, se comparado com os valores anteriores, e os detectores de *Friedman* e *Gossip* apresentaram comportamento semelhante e os melhores valores. Por outro lado, o detector baseado em *cluster* e o detector proposto pelo *Hutle* melhoraram o seu desempenho. Em uma análise, o comportamento desses dois algoritmos tiveram uma melhora mais significativa conforme o aumento do alcance de transmissão se comparado com o algoritmo *Gossip* e o algoritmo do *Friedman* que apresentaram um comportamento semelhante mesmo com a variação do alcance de

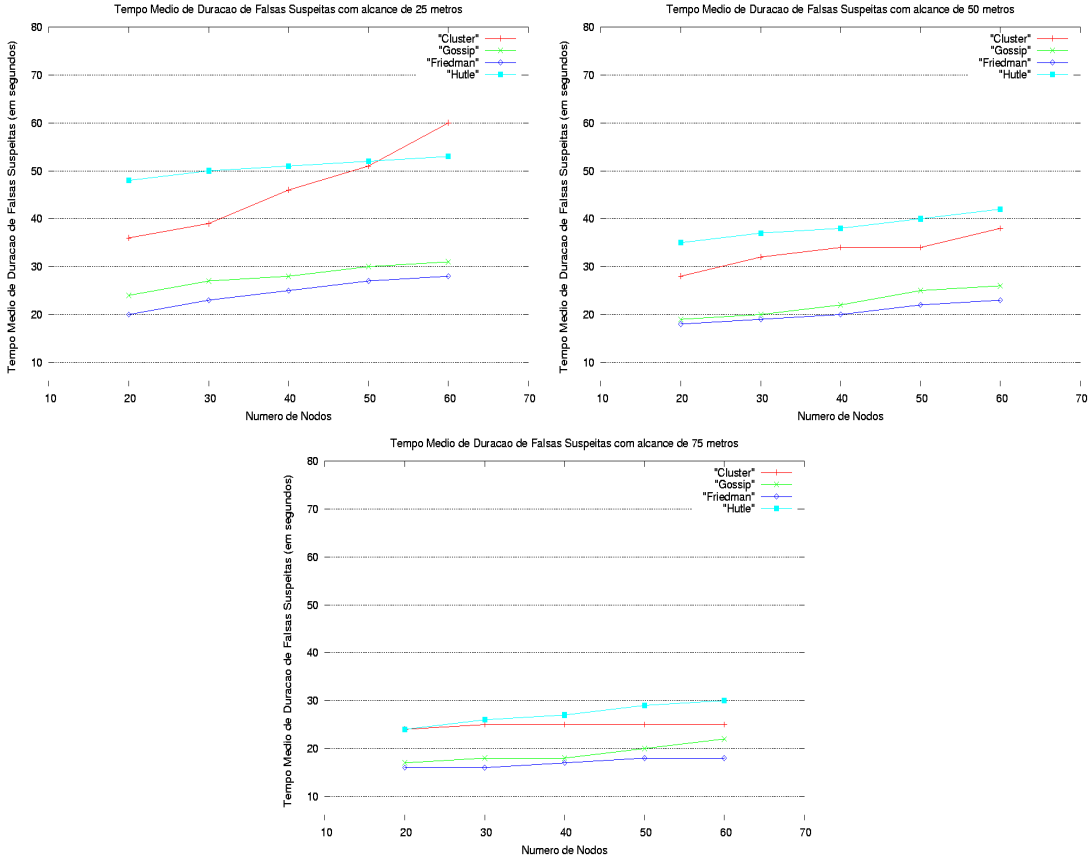

Figura 4. Tempo médio de duração de falsas suspeitas com alcances de transmissão dos nodos de 25, 50 e 75 metros.

transmissão.

3.2.3. Tempo de Detecção

O tempo de detecção mede o tempo para que todos os nodos suspeitem de um nodo defeituoso. Quanto menor for o valor do T_D, mais rápido o detector tomará consciência dos nodos defeituosos. Na simulação, um defeito é simulado fazendo o parar de enviar as mensagens (*broadcasts*). A figura 5 mostra os valores para o T_D com um alcance de transmissão de 25, 50 e 75 metros.

Os detectores de defeitos baseados no algoritmo *Gossip* apresentaram um tempo de detecção praticamente constante, sem grandes variações para um alcance de 25 metros. Especificamente, o T_D no algoritmo *Gossip* se mostrou dependente do valor do $T_{cleanup}$. O algoritmo proposto pelo *Hutle* apresentou os melhores resultados neste alcance. O detector baseado em *cluster* sofreu um aumento no seu T_D com 50 e 60 nodos.

Para 50 metros o algoritmo *Gossip* e o algoritmo do *Friedman* mantiveram os mesmos valores em comparação com o gráfico do alcance de transmissão de 25 metros. Por outro lado, o detector baseado em *cluster* apresentou uma redução em seus tempos médios. Isso porque com um alcance de transmissão maior, o número de reafiliações é menor. Isto é, menos nodos se movem de um *cluster* para outro. Portanto, menor a

probabilidade de erros e menor o tempo de detecção. O algoritmo proposto pelo *Hutle* novamente obteve os melhores resultados e se mostrou constante mesmo com o aumento no número de nodos.

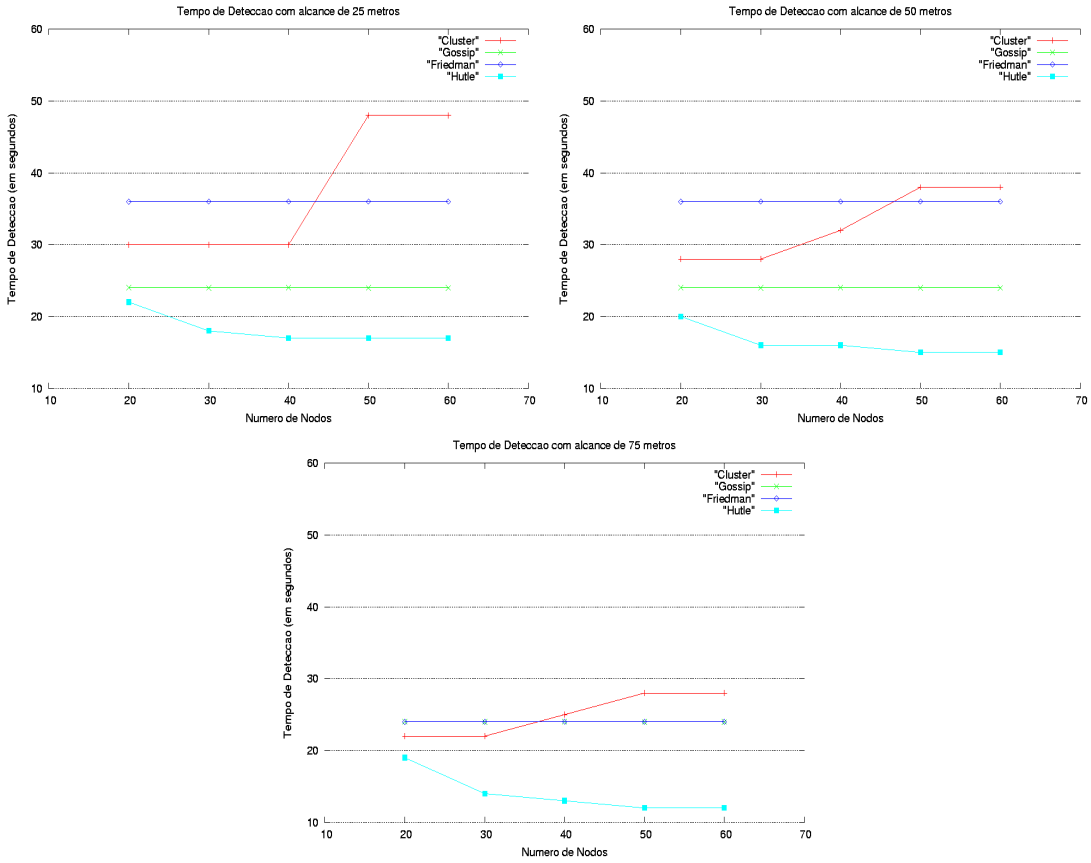

Figura 5. Tempo de detecção com alcances de transmissão dos nodos de 25, 50 e 75 metros.

Para um alcance de 75 metros o algoritmo do *Friedman* sofreu uma diminuição significativa em seus valores, apresentando os mesmos resultados do algoritmo *Gossip*, que se manteve. Os detectores baseado em *cluster* e de *Hutle* também obtiveram valores melhores. Em síntese, observa-se que o T_D do *Gossip* independe do alcance de transmissão, e que o algoritmo do *Hutle* apresenta os melhores valores para o T_D, embora sofra pouca variação com os aumentos no alcance de transmissão e no número de nodos.

3.2.4. Número de *Broadcasts*

A tabela 2 mostra o número total de *broadcasts* para o algoritmo *Gossip*, para os detectores propostos pelo *Hutle* e *Friedman* e para o detector baseado em formação de *cluster*. O algoritmo *Gossip* e o algoritmo proposto pelo *Friedman* possuem o mesmo número de *broadcasts* mesmo com a variação no alcance de transmissão, isso porque o envio de mensagens não depende do alcance e sim do intervalo entre o envio de duas mensagens consecutivas. Já o algoritmo proposto pelo *Hutle*, sofre um aumento no número de *broadcasts* conforme o alcance de transmissão também aumenta. Isso se deve ao fato de

que o algoritmo envia, a cada intervalo de tempo, um *broadcast* para cada vizinho que tem uma distância mínima (um *hop*). Dessa forma, com o aumento do alcance de transmissão um nodo passa a ter mais vizinhos com a distância mínima e conseqüentemente envia um maior número de *broadcasts*. Por fim, no detector de defeitos baseado em *cluster*, o número de *broadcasts* sofre uma pequena queda com o aumento do alcance de transmissão. A explicação para isso é que com um alcance de transmissão maior, menos *cluster* são formados e conseqüentemente um menor número de *clusterheads*, portanto menos mensagens de atualizações dos *clusters* são emitidas pelos *clusterheads*.

Tabela 2. Número de *broadcasts* para os detectores de defeitos com variação no alcance de transmissão e número de nodos.

Detector de Defeitos	20 nodos	30 nodos	40 nodos	50 nodos	60 nodos
Cluster 25 metros	5285	7890	10469	13099	16059
Cluster 50 metros	5245	7875	10444	13069	15802
Cluster 75 metros	5225	7828	10424	13019	15703
Gossip	3000	4500	6000	7500	9000
Friedman	3000	4500	6000	7500	9000
Hutle 25 metros	6667	13294	14587	26531	50372
Hutle 50 metros	6797	14110	20016	32456	52404
Hutle 75 metros	7700	15536	29835	36542	54790

4. Conclusão e Trabalhos Futuros

Este trabalho apresentou uma avaliação entre os detectores de defeitos baseados no algoritmo *Gossip* (*Hutle* e *Friedman*), juntamente com o próprio algoritmo *Gossip* e um detector baseado em *cluster*. Os detectores foram comparados em relação ao número de *broadcasts*, tempo médio de recorrência ao erro (T_{MR}), tempo médio de duração de falsas suspeitas (T_M) e tempo de detecção (T_D).

Como resultado da comparação, observa-se que o algoritmo proposto pelo *Hutle* apresentou o melhor comportamento para o T_{MR} e T_D, o que significa que para uma aplicação onde a velocidade para detectar um erro e a recorrência ao erro sejam os fatores mais relevantes, este algoritmo se torna a melhor opção. Porém foi o detector que mais emitiu *broadcasts*, ou seja, o que tende a consumir mais energia. Observou-se que o detector de defeitos baseado em *cluster* também obteve bons resultados para o T_{MR}, principalmente quando o rádio tem maior alcance, mas não demonstrou competitividade em outras métricas. Por outro lado, em um ambiente onde a energia consumida é o fator mais importante, o detector do *Friedman* e o *Gossip* são os que apresentaram menores números no envio de *broadcasts* e conseqüentemente um menor uso do rádio. Além do baixo consumo, o algoritmo do *Friedman* também apresentou bons resultados para o T_M. O detector baseado em *cluster* teve um rendimento intermediário em preticamente todas as métricas. Em síntese, em um ambiente onde a qualidade de serviço do detector é o fator mais relevante, o detector proposto pelo *Hutle* se mostrou a melhor opção. Mas em um ambiente onde o consumo de energia é essencial, o algoritmo do *Friedman* ou o detector baseado em *cluster* podem ser boas alternativas.

Como seqüência a este trabalho estamos investigando um novo algoritmo que contemple adequadamente os requisitos de mobilidade e consumo.

Referências

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

Chen, W., Toueg, S., and Aguilera, M. K. (2002). On the quality of service of failure detectores. *IEEE Transactions On Computer*, 51(2):561–580.

Felber, P., Défago, X., Guerraoui, R., and Oser, P. (1999). Failure detectors as first class objects. In *Proceedings of the International Symposium on Distributed Objects and Applications (DOA'99)*, pages 132–141, Edinburgh, Scotland.

Friedman, R. and Tcharny, G. (2005). Evaluating failure detection in mobile ad-hoc networks. *Int. Journal of Wireless and Mobile Computing*.

Gerla, M. and Tsai, J. (1995). Multicluster, mobile, multimedia radio network. *Journal of Wireless Networks*, 1(3):255–265.

Gracioli, G. and Nunes, R. C. (2006). Detectores de defeitos para redes wireless ad hoc. *Anais - IX Escola Regional de Redes de Computadores (ERRC)*.

Heidemann, J. S., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., and Ganesan, D. (2001). Building efficient wireless sensor networks with low-level naming. In *Symposium on Operating Systems Principles*, pages 146–159.

Hutle, M. (2004). An efficient failure detector for sparsely connected networks. *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2004), Innsbruck, Austria*.

Jalote, P. (1994). *Fault tolerance in distributed systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Johnson, D. B. and Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers.

Mateus, G. R. and Loureiro, A. A. (2005). *Introdução à computação móvel*. 2ª edition.

Pereira, M. R., l. de Amorim, C., and de Castro, M. C. S. (2003). Tutorial sobre redes de sensores. Cadernos do IME UERJ- Série Informática - Vol 14 - Junho 2003 - Disponível em http://www.ime.uerj.br/cadernos/cadinf/vol14/ - Último acesso em Fevereiro de 2007.

Renesse, R. V., Minsky, Y., and Hayden, M. (1998). A gossip-style failure detection service. Technical Report TR98-1687.

Subramaniyan, R., Raman, P., George, A. D., and Radlinski, M. (2005). Gems: Gossip-enabled monitoring service for scalable heterogeneous distributed systems. Technical Report HCS Lab., Depto. of Electrical and Computer Engineering, Univ. of Florida.

Tai, A. T. and Tso, K. S. (2004). Failure detection service for ad hoc wireless networks applications: A cluster-based approach. Technical Report IAT-302184, IA Tech, Inc., Los Angeles, CA.

Tai, A. T., Tso, K. S., and Sanders, W. H. (2004). Cluster-based failure detection service for large-scale ad hoc wireless network applications. In *Proc. of the 2004 Int. Conf. on Dependable Systems and Networks (DSN'04)*, page 805. IEEE Computer Society.