Um Detector de Defeitos Cumulativo Baseado em uma Abordagem Difusa

Joás E. Souza[1], Fernando Castor[2]
[1]Departamento de Sistemas e Computação
Universidade de Pernambuco (UPE) – Recife, PE – Brazil
[2]Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE - Brazil
jes@dsc.upe.br, castor@cin.ufpe.br

Abstract. *Failure detectors are fundamental components of large-scale distributed systems. A failure detector must perceive node crashes promptly and adapt a wide array of network conditions. In addition, it has to be accurate and avoid wrong suspicions, since the latter result in unnecessary processing and communication. Moreover, failure detectors should, ideally, have modest requirements in terms of memory consumption and processing time. This paper presents an accrual and fuzzy failure detector (ACD). The ACD employs ideas from Fuzzy Logic and outputs a suspicion level on a continuous scale, instead of simply considering a monitored process as "correct" or "suspect". Furthermore, the ACD requires less resources than other accrual failure detectors. Our experiments show that, in some scenarios, the ACD performs better than well-known failure detectors described in the literature and, in others, has a comparable performance.*

Resumo. *Detectores de defeitos são componentes fundamentais de sistemas distribuídos de grande escala. Um detector de defeitos precisa detectar quebras de nós rapidamente, adaptando-se às condições variáveis da rede, além de evitar o excesso de falsas suspeitas, que geram processamento e comunicação desnecessários. Além disso, é desejável que um detector de defeitos tenha requisitos modestos em termos de custo de memória e tempo de processamento. Este artigo apresenta um detector de defeitos cumulativo e difuso (DCD). O DCD utiliza conceitos da Lógica Fuzzy (Difusa) e atribui um nível de suspeita aos processos monitorados em uma escala contínua, ao invés de considerá-lo apenas "correto" ou "suspeito". Adicionalmente, o DCD exige menos recursos do que outros detectores cumulativos. Experimentos mostram que, em alguns cenários, o DCD tem um desempenho melhor que alguns dos principais detectores de defeitos descritos na literatura, enquanto em outros, apresenta um desempenho comparável.*

1. Introdução

Detecção de defeitos é um serviço básico de infra-estruturas para a construção de sistemas distribuídos de grande escala e está presente em diversas aplicações importantes. Exemplos incluem jogos eletrônicos multi-jogador, serviços de mensagens

instantâneas, vídeo-conferência e computação em grade e em nuvem. A detecção rápida de defeitos é fundamental em sistemas críticos, que precisam ser tolerantes a falhas. É imprescindível que esses sistemas sejam capazes de perceber defeitos o quanto antes para que alguma medida de recuperação seja executada evitando que interrompam o fornecimento de seus serviços. Detectores de defeitos são componentes de sistemas distribuídos tolerantes a falhas que monitoram mensagens (*heartbeats*) entre processos e, através delas, estipulam limites de tempo de espera pelo próximo *heartbeat* (*timeouts*) para considerar que um processo monitorado falhou, sendo que a estimativa precipitada ou retardada acarretará em ineficiência e custo operacional desnecessário (processamento e memória). A implementação desses detectores dependerá do modelo de sistema distribuído que pode ou não ser monitorado com algum tipo de sincronização.

Considerando uma quantidade razoável dos últimos *heartbeats* obtidos por um detector de defeitos, é simples estimar o tempo de chegada do próximo. Basta obter a média dos últimos tempos entre *heartbeats* recebidos e acrescentar a essa média o tempo do maior intervalo de atraso (*delay*) obtido [Rachid and Luís (2006)]. Alguns detectores adaptativos funcionam de forma semelhante, incrementando ou decrementando o *timeout* conforme os últimos atrasos observados. Abordagens para melhorar a precisão dos detectores frente às intermitências da rede são as mais diversas. Vão desde o ajuste de *timeout* em função de falsas suspeitas ou últimos atrasos, peculiar dos detectores adaptativos, até a ausência do *timeout* explícito, estabelecendo um limite de espera em função da probabilidade de chegadas considerando distribuição cumulativa normal [Hayashibara, Défago, Yared, and Katayama (2004)] ou função exponencial [Xiong, Défago (2007)], peculiar dos detectores cumulativos.

Esse trabalho propõe um detector capaz de se sintonizar dinamicamente conforme o comportamento do ambiente onde está o processo monitorado, usando poucos recursos de processamento e memória, mas buscando o máximo possível de precisão (*accuracy*) e a menor quantidade possível de falsas suspeitas. O *Detector Cumulativo Difuso* (DCD) proposto por esse trabalho é um detector de defeitos não confiável [Chandra and Toueg (1996)] e segue um modelo assíncrono. Isso singifica que não é possível fazer qualquer suposição sobre os tempos entre chegas de *heartbeats*, tornado bastante difícil a tarefa de diferenciar processos lentos e processos realmente falhos. Além da capacidade de avaliar o nível de carga da rede, adotando uma estratégia agressiva de detecção em redes relativamente estáveis e uma estratégia conservadora em redes instáveis, o DCD também atende a todas as propriedades de um detector cumulativo descritas por [Défago, Urbán, Hayashibara, Katayama (2005)]. Em outras palavras, o detector não apenas julga de forma binária se um processo é suspeito ou confiável, mas é capaz de atribuir um nível de suspeição contínuo que atende configurações de QoS para determinadas aplicações que tratem de forma diferenciada determinados processos com maior ou menor probabilidade de falhas. Daí surge a afinidade do detector proposto com Lógica Difusa, que estabelece alternativas à lógica binária e, ao invés de considerar algo verdadeiro ou falso, suspeito ou confiável, sugere que um processo seja parcialmente suspeito e parcialmente confiável.

Este artigo está dividido em seis seções. A Seção 2 apresenta uma visão geral sobre detecção de defeitos, com especial ênfase nos detectores de defeitos adaptativos, desde os binários aos cumulativos. A Seção 3 introduz alguns conceitos básicos da

Lógica Difusa, úteis para compreender o detector proposto, apresentado na Seção 4. . A Seção 5 descreve a avaliação do DCD. Essa avaliação consistiu em várias simulações envolvendo *traces* reais e comparou o DCD com diversos outros detectores adaptativos descritos na literatura. Por fim, a Seção 6 conclui o trabalho e descreve os próximos passos a serem dados em trabalhos futuros.

2. Detectores de Defeitos Adaptativos

A detecção de defeitos é fundamental para garantir tolerância a falhas em sistemas distribuídos. A detecção é necessária na resolução de diversos problemas importantes como consenso [Gartner (1999)] e difusão atômica de informação [Xavier Défago, André Schiper, Péter Úrban (2004)]. Um importante resultado teórico na área de sistemas distribuídos [Fischer, Lynch and Peterson (1985)] mostra que é impossível resolver o problema de consenso no modelo assíncrono de interação, ou seja, quando não há um limite superior de tempo para realizar uma ação, como enviar ou processar uma mensagem. Um trabalho posterior mostrou que esse resultado decorre da impossibilidade de detectar processos defeituosos de maneira confiável, já que não há como diferenciar a falha de um processo de uma transmissão muito lenta em sistemas assíncronos [Chandra and Toueg (1996)]. Considerando essa impossibilidade, surgem dois grandes problemas: processos falhos não suspeitos (falsas negativas) e processos suspeitos não falhos (falsas positivas). Normalmente, abordagens práticas lidam com o primeiro problema determinando um tempo limite dentro do qual os processos precisam responder a requisições. Se esse tempo se passa sem que uma resposta chegue, o processo é considerado falho. Essa solução resolve o primeiro problema, mas cria abertura para o segundo. Consequentemente, abordagens para detecção de defeitos buscam atingir uma alta precisão na detecção com um baixo tempo médio de detecção.

Desde 1984 alguns algoritmos já consideravam a identificação de processos falhos em sistemas distribuídos [Schneider, Gries, Schlichting (1984)]. Em 1991 já se monitorava o tempo de transmissão de processos para detectar falhas. Alguns anos depois, foi proposto o conceito de detector de defeitos não confiável em sistemas distribuídos. O trabalho seminal sobre detectores de defeitos não confiáveis [Chandra and Toueg (1996)] é a fonte de grande parte da terminologia da área de detecção de defeitos. Esse trabalho define detectores de defeitos não confiáveis no contexto de sistemas distribuídos assíncronos com canais de comunicação confiáveis, onde os processos podem falhar, mas não se recuperam após a falha. Cada processo possui um módulo detector de defeitos e uma lista com os processos suspeitos de terem falhado. Esses detectores de defeitos não são confiáveis porque podem fazer suspeitas incorretas sobre quais processos estão ativos e quais falharam, isto graças à ausência limites superiores para os tempos de processamento e de transmissão de mensagens.

Para evitar a definição de detectores de defeitos baseada em particularidades de implementação, detectores de defeitos são classificados de acordo com as propriedades abstratas de completude (*completeness*) e precisão (*accuracy*) [Chandra and Toueg (1996)]. A propriedade de completude indica a capacidade de algum ou todos os processos suspeitarem, de forma permanente, de todos os processos que falharam. A propriedade de precisão indica quando algum ou todos os processos ativos podem ser suspeitos de ter falhado. Essas propriedades podem ser combinadas de diversas maneiras para se produzir diferentes classes de detectores.

Alguns detectores, como o proposto por Bertier *et al* [Bertier, Marin and Sens (2002)], atribuem uma suspeição binária (suspeito ou confiável) ao processo monitorado. Essa abordagem se fundamenta em idéias semelhantes aos serviços de garantia de entregas de pacotes do protocolo TCP, propostos por Jacobson [Jacobson (1988)]. Neles estima-se, através de uma fórmula, o tempo de chegada dos próximos *heartbeats*, dando maior significância aos tempos de chegada mais recentes de forma intuitiva ou heurística.

Outros, como o proposto por Chen *et al* [Chen, Toueg and Aguilera (2000)], são baseados na análise probabilística do tráfego da rede. O protocolo usa amostras dos tempos de chegada mais recentes para estimar o tempo de chegada dos próximos *heartbeats*. O tempo Δ_t é obtido através dessa estimativa somada a uma constante de segurança α e atualizado após a chegada de cada *heartbeat*. A margem de segurança é baseada em configurações de *QoS*. Caso o processo monitor não receba esse *heatbeat* dentro do limite de espera calculado, o processo monitorado passa a ser considerado suspeito. Esse detector possui uma versão que considera relógios sincronizados e outra que considera os não-sincronizados.

Há ainda detectores que não fazem medições de tempo nem tampouco utilizam o conceito de *timeout* para suspeitar ou não de um determinado processo [Sens, Arantes, Bouillaguet, Simon, Greve (2008)]. Sens *et al.* propõem um detector para dispositivos móveis que envia mensagens em *broadcast* e espera respostas de outros processos que receberam essas mensagens. Dessa forma os processos passam a se perceber e continuam reenviando e recebendo mensagens informando suas listas de processos "suspeitos" e "provavelmente suspeitos" uns para os outros. Esse detector difere do DCD porqueeste último utiliza tempos entre chegadas de *heartbeats* para definir o nível de suspeição dos processos monitorados e funciona como um componente isolado, sem a necessidade de se comunicar com qualquer processo além daqueles que monitora.

Alguns trabalhos [Hayashibara, Défago, Yared and Katayama (2004)], [Satzger, Pietzowski, Trumler and Ungere (2007)] defendem a idéia de que, tendo em vista a falta de precisão inerente à tarefa de detectar nós defeituosos, não é razoável que várias abordagens para detecção de defeitos forneçam, saídas binárias, indicando apenas se um nó é suspeito ou não de ter falhado. Hayashibara *et al* [Hayashibara, Défago, Yared and Katayama (2004)] propuseram então um novo tipo de detector de defeitos que utiliza níveis variados de suspeição. Tais detectores, ao invés de considerarem nós monitorados simplesmente como suspeitos ou não, atribuem um nível contínuo de suspeição que se baseia nos intervalos entre mensagens recebidas anteriormente oriundas do mesmo processo monitorado. Fica a cargo da aplicação que utiliza o detector determinar que medida será tomada em resposta a determinados níveis de suspeição. Detectores de defeitos com essas características foram batizados de detectores de defeitos cumulativos (a*ccrual failure detectors*). Esses detectores armazenam os tempos entre *n* chegadas consecutivas de *heartbeats*, onde *n* é um parâmetro do detector (janela), e se utilizam desses dados para avaliar o ambiente de transmissão e atribuir níveis de suspeição ao processo monitorado.

Apesar do conceito detectores de defeitos cumulativos ter sido introduzido recentemente, já há alguns trabalhos que propõem detectores de defeitos baseados nessa abordagem. Por exemplo, Hayashibara *et al* [Hayashibara, Défago, Yared and

Katayama (2004)] propuseram o Detector de Defeitos φ . Esse detector utiliza uma fila que guarda os *n* mais recentes intervalos entre *heartbeats*. O detector supõe que os tempos entre chegadas de *heartbeats* podem ser modelados pela distribuição normal e, com base nisso, estima a probabilidade P(*t*) do próximo *heartbeat* ainda chegar dado que já transcorreram *t* unidades de tempo desde que o último *heartbeat* foi recebido. As informações guardadas pelo detector são atualizadas a cada chegada de um *heartbeat*. Detectores adaptativos cumulativos que consideram as chegadas de heartbeats eventos aleatórios são eficientes nos ambientes que têm esse comportamento. Apesar disso, é importante salientar que a chegada de heartbeats não é necessariamente um evento aleatório, mas estocástico, visto que os intervalos entre heartbeats consecutivos não podem ser considerados eventos independentes.

[Satzger *et al*. [Satzger, Pietzowski, Trumler and Ungere (2007)] propuseram um detector de defeitos cumulativo que também faz uso de uma fila onde são guardados os intervalos entre *heartbeats*. Quando o detector está à espera de um *heartbeat*, o tempo decorrido desde a chegada do *heartbeat* anterior até o momento atual é comparado aos intervalos armazenados e é contado quantos destes são menores. Dessa forma tem-se o percentual de suspeição de que o próximo *heartbeat* nunca chegará. Esse detector utiliza um fator de escala, similar à margem de segurança usada em abordagens adaptativas [Chen, Toueg and Aguilera (2000)], para reduzir o número de suspeitas incorretas.

Nos dois detectores de defeitos cumulativos citados anteriormente há um elemento essencial para definir o quanto o detector é tolerante, o limiar, ou *threshold,* do detector. O limiar de um detector cumulativo é um valor numérico que define o limite de espera, ou seja, que valor o detector precisa assumir para que um processo monitorado passe a ser considerado suspeito. Detectores cumulativos permitem que diversos limiares sejam configurados e que a cada um deles seja associada uma ação, por exemplo, a restauração de um ponto de recuperação armazenado anteriormente. Com isso, diferentes estratégias podem ser empregadas depndendo do quão grande é o nível de suspeita de que um processo falhou. No trabalho de Hayashibara et al.[Hayashibara, Défago, Yared and Katayama (2004)], o limiar corresponde ao expoente negativo do valor da probabilidade de um *heartbeat* ainda chegar, ou seja quanto maior for o limiar escolhido, menor será o valor da probabilidade da chegada de um *heartbeat* tolerada. Já no detector proposto por Satzger et al.[Satzger, Pietzowski, Trumler and Ungere (2007)], o limiar identifica em que posição o tempo atual estaria num conjunto ordenado dos mais recentes intervalos entre chegadas de *heartbeats*, caso essa posição seja maior do que o limiar o detector considera o processo monitorado como suspeito.

3. Breve Introdução à Lógica Difusa

A Teoria de Conjuntos Fuzzy (difusos) foi concebida com o objetivo de fornecer um ferramental matemático para o tratamento de informações de caráter impreciso ou vago. A Lógica Difusa, baseada nessa teoria, foi inicialmente construída a partir dos conceitos já estabelecidos de lógica clássica; operadores foram definidos à semelhança dos tradicionalmente utilizados e outros foram introduzidos ao longo do tempo, muitas vezes por necessidades de caráter eminentemente prático [Tanscheit, R. (2003)].

Na teoria clássica dos conjuntos, o conceito de pertinência de um elemento a um conjunto fica bem definido. Dado um conjunto A em um universo X, os elementos deste universo simplesmente pertencem ou não pertencem àquele conjunto. Zadeh propôs uma caracterização mais ampla, generalizando a função característica de modo que ela pudesse assumir um número infinito de valores no intervalo [0,1]. Um conjunto difuso A em um universo X é definido por uma função de pertinência e representado por um conjunto de pares ordenados

$$A = \{\mu_A(x)/x\}\ x \in X$$

onde $\mu_A(x)$ indica o quanto x é compatível com o conjunto A. Um determinado elemento pode pertencer a mais de um conjunto difuso, com diferentes graus de pertinência [Tanscheit, R. (2003)].

O conjunto suporte de um conjunto difuso A é o conjunto de elementos no $x \in X$ para os quais $\mu_A(x) > 0$. Um conjunto difuso cujo suporte é um único ponto x' com $\mu_A(x') = 1$ A é chamado de conjunto unitário difuso ou *singleton*. Assim, um conjunto difuso também pode ser visto como o mapeamento do conjunto suporte no intervalo [0,1], o que implica em expressar o conjunto difuso por sua função de pertinência [Tanscheit, R. (2003)].

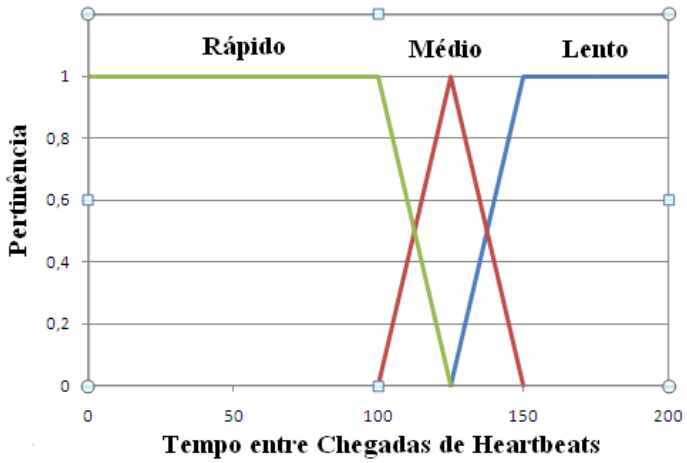

Figura 1: Funções de pertinência para a variável Tempo entre Chegadas

Funções de pertinência podem ser definidas a partir das mais diversas suposições, mas é comum fazer-se uso de funções de pertinência padrão, como, por exemplo, as de forma triangular, trapezoidal e Gaussiana. Em aplicações práticas as formas escolhidas inicialmente podem sofrer ajustes em função dos resultados observados [Tanscheit (2003)]. A Figura 1 representa possíveis funções de pertinência. Verifica-se que um determinado intervalo pode fazer parte de duas funções de pertinência possuindo maior grau de pertinência a uma do que a outra.

4. Uma Proposta Difusa para Detecção de Defeitos

Os detectores adaptativos não cumulativos e cumulativos têm como principal diferença a forma como suspeitam dos processos monitorados. O primeiro deles atribui suspeita de forma binária (suspeito ou confiável) como a Lógica Tradicional (Aristotélica) e o segundo de forma contínua e nebulosa, sem limites preestabelecidos, como a Lógica

Difusa (Fuzzy) [Zadeh, L.A. (1965)]. Tendo em vista essas similaridades esta seção apresenta um novo detector de defeitos cumulativo que lança mão de idéias oriundas da lógica difusa, o DCD.

O detector proposto leva em consideração a importância de evitar falsas suspeitas e detectar processos defeituosos rapidamente em cenários diversos. O DCD não pré-supõe o comportamento da chegada dos *heartbeats*, se esses trafegam por uma LAN, WAN, sem fio ou com fio. Ele gradualmente se adapta às condições da rede. Quando a rede é estável, com tempos entre chegadas de *heartbeats* apresentando uma variância pequena, o nível de suspeita dos processos monitorados cresce rapidamente, o que diminui o tempo de detecção. Em redes instáveis, com uma alta variância, o DCD adota uma abordagem conservadora, incrementando níveis de suspeita mais lentamente.

De um modo geral, o detector prioriza um número de falsas positivas baixo em detrimento de um tempo de detecção maior. Plataformas de middleware reais em domínios como grades computacionais aceitam longos tempos de detecção portanto que falsas positivas ocorram raramente. Por exemplo, o InteGrade [Goldchleger et al. (2004)] e o OurGrid [Cirne et al. (2006)] usam detectores simples que esperam, respectivamente, 30 minutos e 5 minutos (em suas configurações padrão) antes de considerar que um processo falhou. Ao forçar a ocorrência de falsas positivas em cada um deles, foi possível perceber que ambos tornam-se inconsistentes e precisam ser reiniciados. Em outras palavras, aplicações reais frequentemente não estão prontas para lidar com falsas positivas e precisam de precisão na detecção de defeitos.

O DCD também parte do pressuposto de que recursos como processamento e memória devem ser economizados em algumas aplicações distribuídas como os sistemas ubíquos e as grades computacionais oportunistas. O DCD leva em consideração os últimos *heartbeats*, assim como os principais detectores cumulativos existentes, mas não os armazena, nem tampouco utiliza fórmulas complexas, como o Detector de Defeitos φ [Hayashibara, Défago, Yared and Katayama (2004)]. Essa economia de recursos decorre da abordagem difusa usada em seu desenvolvimento.

4.1. Uma Abordagem Ingênua

Inicialmente, foi desenvolvido um detector de defeitos inspirado na Lógica Difusa com várias funções de pertinência que definiam os conjuntos difusos. As funções de pertinência definidas se baseavam nos *heartbeats* recebidos e, conforme o tempo de chegada, os **classificava** como "*heartbeats* lentos", "*heartbeats* normais" e "*heartbeats* rápidos", levando em consideração o tempo de detecção médio (TDM) e as falsas positivas (FP). A partir dessa classificação, **medidas reativas** responsáveis por ajustar a tolerância do detector a atrasos foram criadas, também definidas através de funções de pertinência. Essas medidas sugeriam ações como: "aumentar *tolerância*", "*não mudar*", "diminuir *tolerância*". A combinação dessas funções produz um detector adaptativo capaz de **atribuir suspeita** de forma contínua e em linguagem natural: "muito confiável", "confiável", "suspeito", "muito suspeito".

O maior desafio no projeto desse detector inicial era a definição das funções de pertinência peculiares às três etapas do processo (classificação dos intervalos entre chegadas, reação, através da modificação do limite superior esperado para a chegada de novos *heartbeats* e atribuição do nível de suspeita) conforme suas características, de

forma justa e representativa, definindo que forma e dimensões teriam tais funções. Além desse, outro desafio era ajustar todas essas funções dinamicamente a fim de que representassem o estado do meio de transmissão de dados a cada instante. Em particular, em áreas de interseção entre os conjuntos difusos, torna-se difícil definir como o detector deve reagir, se deve tornar-se menos tolerante a atrasos, mais tolerante ou não mudar sua atitude. Essa dificuldade é exacerbada pelo dinamismo de redes sem fio e de grade área. Como conseqüência dessas dificuldades, esse detector ingênuo adaptava-se muito lentamente às condições de uma rede de grande área, apesar de várias configurações de funções de pertinência, conjuntos difusos e reações terem sido avaliadas. Ao final, concluiu-se que o uso de vários conjuntos difusos não seria apropriado para lidar com o alto grau de dinamismo de um sistema distribuído.

4.2. Detector Proposto

O principal resultado da experiência descrita na seção anterior foi o projeto de um novo detector de defeitos que não possui vários conjuntos difusos, mas que concentra num único conjunto os tempos dos *heartbeats* recebidos. Uma vantagem óbvia dessa abordagem é que uma função de pertinência é suficiente para definir a maneira como o detector deve reagir, inclusive nos casos de variações bruscas nos intervalos entre chegadas de *heartbeats*. O funcionamento do DCD é bastante simples e pode ser resumido da seguinte maneira. Ao receber um *heartbeat*, ele verifica o quão perto dos limites (inferior e superior) do seu conjunto de pertinência o tempo entre a chegada deste *heartbeat* e o anterior se encontra. Quanto mais perto estiver do limite inferior, mais é reduzida sua tolerância a intervalos longos, ou seja, menores tornam-se seus limites superior e inferior. Da mesma forma, quanto mais perto do limite superior, mais é aumentada a tolerância do detector. Intervalos localizados fora do conjunto de pertinência e maiores que seu limite superior podem ou não resultar no processo monitorado ser considerado suspeito, dependendo dos limiares estabelecidos. Ao mesmo tempo, resultam em aumentos expressivos no limite superior do conjunto.

DCD se baseia em duas variáveis principais que, juntamente com os tempos entre chegadas e o limiar definido pelo usuário, definem o quanto o detector espera pelo próximo *heartbeat* antes de considerar que o processo monitorado falhou (ou seja, sua reação): *lim_inferior* e *lim_superior*. A variável *lim_inferior* é modificada em duas situações. Na primeira, um intervalo entre chegadas é menor que esse limite. Neste caso, o *lim_inferior* passa a assumir o valor desse intervalo, o que diminui a tolerância do detector a intervalos longos. Na segunda, um intervalo *ival* entre chegadas é maior que a média aritmética entre *lim_superior* e *lim_inferior*, conforme mostrado a seguir:

$$ival > \frac{lim\_superior + lim\_inferior}{2} \tag{1}$$

Quando isso ocorre, lim_inferior muda conforme a expressão

$$lim\_inferior = lim\_inferior_0 + \left(\frac{lim\_superior - lim\_inferior_0}{V} \right) \tag{2}$$

onde $lim\_inferior_0$ é o valor de *lim_inferior* antes da modificação. Na expressão acima, V é a velocidade de ajuste do detector, um parâmetro configurado por seus usuários. A velocidade de ajuste determina o quão lentamente o detector modifica seu conjunto de

pertinência no caso em que o último intervalo entre *heartbeats* registrado é um de seus elementos. Nos experimentos realizados até o momento, foi constatado que quanto maior o valor de *V*, mais tolerante o detector se torna e, consequentemente, menos enganos comete. Valores de *V* na faixa de 500 a 2000 foram os que produziram os melhores resultados.

As duas situações descritas acima ilustram duas características fundamentais do DCD: (i) quando os intervalos são muito curtos (ou muito longos, como será visto a seguir), o DCD ajusta o tamanho do seu conjunto de pertinência de forma agressiva; e (ii) quando os intervalos entre chegadas de *heartbeats* estão dentro do esperado (são elementos do conjunto de pertinência), o detector se adapta suavemente.

O valor de *lim_superior* é modificado de maneira similar a *lim_inferior*, de acordo com a fórmula

$$lim\_superior = lim\_superior_0 + \left(\frac{lim\_superior_0 - lim\_inferior}{V} \right) \quad (3)$$

caso o último intervalo entre chegadas obtido seja menor que a media aritmética entre *lim_superior* e *lim_inferior*. Se o intervalo for maior que essa média, o limite superior recebe um incremento de $\left(\frac{lim\_superior - lim\_inferior}{V} \right)$. Se o intervalo entre os dois últimos *heartbeats* recebidos for maior que *lim_superior*, este último passar a ter um valor igual a esse intervalo, aumentando a tolerância do detector.

O nível de suspeição fornecido pelo DCD é um número igual a *ival* – *lim_superior* , onde *ival* representa o intervalo decorrido desde o recebimento do último *heartbeat* até o momento atual. Um valor negativo para o nível de suspeição significa que, com base no histórico do processo monitorado, ainda é mais provável que um novo *heartbeat* chegue do que que o processo tenha falhado. Um valor positivo indica que é mais provável que uma falha tenha ocorrido.

No DCD, limiares são definidos por valores numéricos $L_{1..N}$, tais que $\forall i \,|\, 1 \le i \le N \bullet L_i > 0$. Quanto maior o valor de um limiar, maior a tolerância do detector a longos intervalos entre *heartbeats*. Nos experimentos realizados até o momento, os valores utilizados para definir limiares variaram entre 0,8 e 1,2. Para um limiar *L,* o DCD considera que um processo é suspeito se

$$ival > L * lim\_superior \quad (4)$$

Neste caso, o detector executa a ação de recuperação associada a *L*.

5. Avaliação

Esta seção descreve os detalhes dos experimentos que foram realizados para avaliar o DCD. A avaliação foi feita a partir de simulação, usando *traces* de redes reais. Inicialmente é descrito o ambiente em que a avaliação foi realizada. Em seguida, a Seção 5.2 apresenta os resultados obtidos pelo detector quando comparado a alguns detectores descritos na literatura. Por fim, na Seção 5.3, o desempenho do detector é

avaliado quando modifica-se o valor de seus dois principais parâmetros, a velocidade de ajuste e o limiar (Seção 4.2).

5.1. Ambiente da Avaliação

Os resultados da avaliação foram obtidos através de simulação. Dessa forma é possível garantir que todos os detectores avaliados serão submetidos às mesmas condições, evitando que um ou outro tenha resultados prejudicados por intermitencias da rede ou dos processos monitorado. Todas as simulações foram executadas no mesmo computador, em condições similares. Os detectores foram comparados em três cenários diferentes: (i) uma rede de grande área intercontinental; (ii) uma rede local sem fio; e (iii) uma rede local sem fio com perdas de pacote induzidas. O primeiro cenário representa uma rede relativamente estável, com alguns momentos pontuais de instabilidade e poucas rajadas, a maior parte delas curta. O *trace* utilizado neste cenário é o mesmo obtido por Hayashibara *et al.* [Hayashibara, Défago, Yared, and Katayama (2004)]. Trata-se de quase seis milhões de tempos das chegadas de *heartbeats*. Esses *heartbeats* foram enviados por um computador localizado no Japão e registrados por outro localizado na Suiça. *Heartbeats* foram enviados a cada 100 ms, durante cerca de uma semana e sem interrupções. A transmissão foi feita através de uma conexão *Internet* comum intercontinental entre os computadores. Através do *traceroute* foi possível perceber que a maioria dos *heartbeats* passava pelos Estados Unidos no seu caminho entre Ásia e Europa.

Para os dois outros cenários, os *traces* foram obtidos atráves de uma organização similar à descrita acima, mas com os dois computadores localizados na mesma edificação (em cômodos distintos) e ligados por uma rede local sem fio aderente ao padrão IEEE 802.11. *Heartbeats* foram enviados pelo transmissor a cada 100 ms por um período de aproximadamente 11 horas, sem interrupções, o que totalizou 400.000 *heartbeats*. Durante a obtenção dos intervalos, uma taxa de perda de pacotes inferior a 0,1% foi observada. Isso motivou a obtenção do terceiro cenário. Neste último, *heartbeats* foram enviados e registrados da mesma forma que no segundo cenário. Entretanto, cada intervalo registrado tinha uma probabilidade de 1% de ser descartado. O *trace* resultante desse cenário, consequentemente, reflete uma rede com um grau atipicamente alto de instabilidade.

A comparação entre detectores foi feita em termos de duas métricas de qualidade, tempo médio de detecção e taxa de enganos (número de falsas positivas por segundo). Esses atributos são usados em diversos outros trabalhos para comparar diferentes detectores de defeitos. A comparação envolveu os detectores propostos por Chen et al. [Chen et al. (2000)], Hayashibara et al. [Hayashibara et al. (2004)], Satzger et al. [Satzger et al. (2007)] e Sotoma e Madeira [Sotoma e Madeira (2001)].

5.2. Comparação dos detectores

Nas comparações entre os principais detectores e o DCD, este último foi configurado com um limiar de 1 e velocidade de ajuste 1750. A Seção 5.3 mostra os efeitos de variar esses parâmetros no desempenho do detector. Os outros detectores foram executados com as configurações recomendadas nos trabalhos que os propõem.

A Figura 2 mostra o resultado da comparação entre os detectores no primeiro cenário, a rede de grande área. A taxa de enganos é apresentada em escala logarítmica. O tempo de chegada médio de *heartbeats* neste cenário foi de 103,5ms e desvio padrão de 0,19ms. Para esta configuração, o DCD apresentou um desempenho superior a todos os outros detectores. Seu tempo detecção se manteve baixo, com a média ficando em aproximadamente 117 ms. Apesar desse baixo tempo de detecção, o DCD também apresentou a menor taxa de enganos. A segunda menor taxa de enganos, obtida pelo detector de Sotoma e Madeira, foi aproximadamente 11% maior que a do DCD, enquanto seu tempo de detecção médio foi praticamente igual (ligeiramente maior). Essa comparação mostra que, num cenário real de uma rede de grande área, o detector proposto atinge seu objetivo, que é adaptar-se às intermitências da rede mantendo um baixo número de falsas positivas e, tanto quanto possível, um baixo tempo de detecção.

A Figura 3 representa os resultados da comparação em uma rede local sem fio. Este cenário apresenta um grau de instabilidade bem maior que a rede de grande área do primeiro cenário. Isso se torna evidente pelas taxas muito mais altas de enganos e tempos médios de detecção maiores. Neste cenário, a média dos intervalos entre *heartbeats* foi de 143,9ms e o desvio padrão foi de 241,1 ms, mais de mil vezes maior que o da rede de grande área. Foi possível observar um grande número de intervalos consideravelmente maiores, em decorrência das intermitências da rede sem fio. Esses intervalos ocorreram com regularidade, o que influenciou o desempenho do DCD.

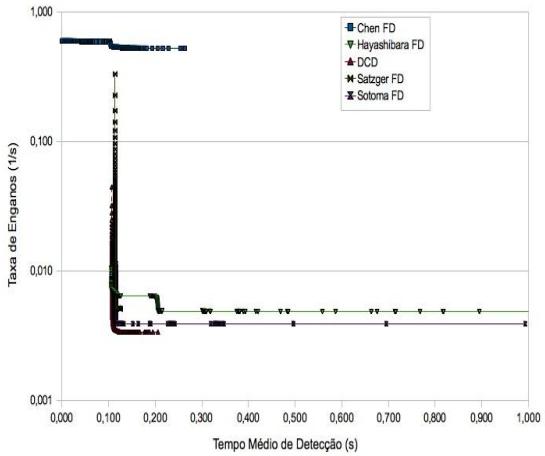

Figura 2. Comparação de detectores de defeitos em uma WAN intercontinental.

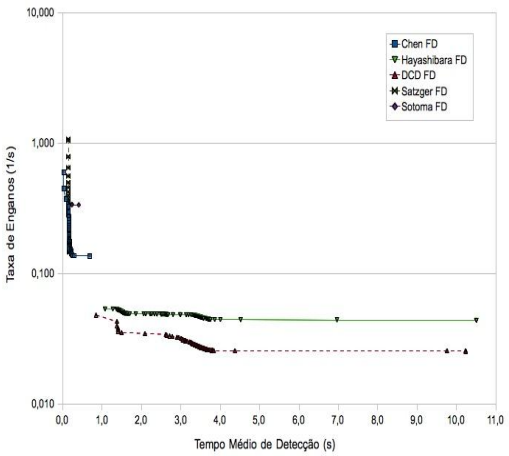

Figura 3. Comparação de detectores de defeitos em uma rede local sem fio.

Enquanto os outros detectores, excetuando-se o de Hayashibara et al., passaram ao largo da intermitência da rede sem fio, o DCD a levou em conta, aumentando sua tolerância a atrasos de forma agressiva. Consequentemente, sem nenhuma configuração adicional, obteve novamente a menor taxa de enganos, quase metade da taxa de enganos do detector de Hayashibara et al., o segundo melhor. Com relação ao detector de Sotoma e Madeira, o segundo melhor em termos de taxa de enganos na rede de grande área, a taxa de enganos do DCD foi mais que 12 vezes menor. O custo desse excelente desempenho, em termos de enganos, foi o elevado tempo médio de detecção, o maior entre todos os detectores avaliados, seguido de perto pelo de Hayashibara et al.

Essa avaliação mostrou duas características importantes da maioria dos detectores avaliados. A primeira é que seus projetos priorizam baixo tempo de detecção em relação à taxa de enganos. A segunda é que esses detectores foram claramente projetados para funcionar bem em certos cenários, em detrimento de outros. Os detectores de Satzger et al. e Sotoma e Madeira tiveram um bom desempenho na rede de grande área mas não se adaptaram tão bem à rede sem fio. Já o de Chen et al. funcionou bem na última, mas não na primeira. Os únicos que se adaptaram bem às duas condições foram o DCD e o de Hayashibara et al.

Por fim, a Figura 4 mostra os resultados obtidos no terceiro cenário. Neste caso, o tempo de chegada médio foi de 161, 2ms, enquanto o desvio padrão foi de 359,4ms. Pela figura, é possível observar que os tempos de detecção médios de todos os detectores aumentaram. Mais uma vez o DCD obteve a menor taxa de enganos, menos que metade da taxa do segundo colocado, o detector de Hayashibara et al. O tempo médio de detecção novamente foi o mais alto, com o detector de Hayashibara et al. em segundo, mas muito próximo.

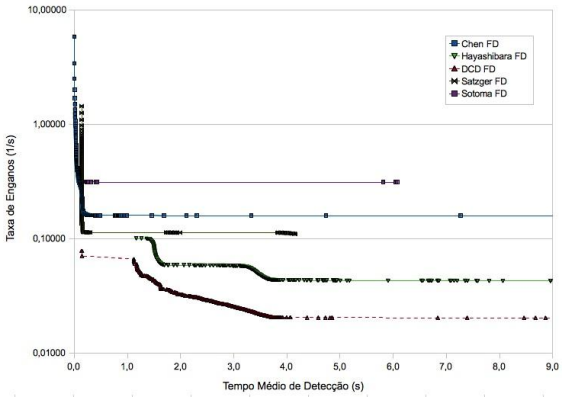

Figura 4. Comparação entre detectores em uma rede sem fio com injeção de perdas.

5.3. Experimentos com ajuste de parâmetros

O detector proposto tem dois parâmetros que podem ser configurados: limiar e velocidade de ajuste (Seção 4.2). Esta seção avalia a influência desses parâmetros no desempenho do DCD e mostra que eles, de fato, modificam seu desempenho. Todos os resultados apresentados decorrem de simulações num ambiente de rede *wireless* instável (o mesmo do terceiro cenário da Seção 5.2), onde ajustes nos parâmetros do detector têm maior probabilidade de ser necessários.

Limiar	**Taxa de enganos (1/s)**	**Tempo de detecção (ms)**
0,95	0,3557	3359,3
1,00	0,0218	3536,8
1,05	0,0138	3713,7
1,10	0,0101	3890,5
1,15	0,0076	4067,3

Tabela 1. Impacto do limiar no desempenho do detector.

Velocidade de Ajuste	Taxa de enganos (1/s)	Tempo de detecção (ms)
50	0,1240	1099,8
500	0,0321	2715,8
1000	0,0258	3201,6
1750	0,0219	3536,8
2500	0,0196	3727,0
5000	0,0155	4065,3
7500	0,0130	4265,6
10000	0,0117	4408,0

Tabela 2. Impacto da velocidade de ajuste no desempenho do detector.

A Tabela 1 mostra que o limiar tem uma grande influência na taxa de enganos. Para um limiar de 1, usado em todos os experimentos até aqui, o DCD apresenta uma taxa de 0,0218 enganos/s. Já para um limiar de 1,15, essa taxa de enganos cai para um valor quase 3 vezes menor. Isso indica que o limiar, conforme usado pelo detector proposto, de fato é útil para torná-lo mais ou menos tolerante a atrasos e, consequentemente, mais ou menos preciso. A Tabela 2 mostra a influência da velocidade de ajuste. Um valor alto para velocidade de ajuste indica que o detector se adapta lentamente quando confrontado com condições de rede estáveis (embora continue se adaptando de fora agressiva quando a rede é instável).

6. Conclusão

Esse trabalho apresenta o Detector Cumulativo e Difuso (DCD), um novo detector de defeitos que agrega idéias da Lógica Difusa [Zadeh (1965)] e propriedades de um Detector Cumulativo [Défago, Urbán, Hayashibara, Katayama (2005)]. Foram realizados experimentos, através de simulação, para comparar o desempenho do DCD com os detectores propostos por diversos outros autores, usando os melhores parâmetros sugeridos nos respectivos artigos. O DCD é um detector que utiliza menos recursos de memória e processamento que os demais comparados nesse artigo e mesmo assim tem uma taxa de enganos menor que as dos demais, com tempo de detecção menor em redes estáveis e comparável aos outros em redes mais instáveis.

Visto que idéias de uma técnica de Inteligência Artificial foram eficientes na detecção de defeitos, pretende-se, em trabalhos futuros, investigar outras técnicas de Inteligência Computacional e Otimização Multi-Objetivo para agregarem valor a sistemas tolerantes a falhas e detectores de defeitos. Outro trabalho futuro consiste em avaliar o DCD em uma rede real, ao invés de usar simulação.

Agradecimentos

Os autores agradecem a Naohiro Hayashibara por ter cedido o *trace* para o cenário da rede de grande área (Seções 5.1 e 5.2). Os autores também gostariam de agradecer aos revisores anônimos pelos vários comentários úteis. Joás é financiado pelo CNPq/Brasil. Fernando é financiado pelo CNPq/Brasil, 308383/2008-7, 481147/2007-1, e 550895/2007-8, e pelo Instituto Nacional de Ciência a Tecnologia para Engenharia de Software, financiado por CNPq e FACEPE, 573964/2008-4 e APQ-1037-1.03/08.

Referências

Bertier, M., Marin, O., Sens P. (2002). Implementation and performance evaluation of an adaptable detector. In Proc. DSN'2002, pages 354-363, Washinton, DC, USA.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. In Journal of ACM, volume 43 Issue 2, pages 225–267.

Chen, W., Toueg, S., and Aguilera, M. K. (2000). On the quality of service of failure detectors. Proc. DSN'2000, pages 561-580, New York, USA.

Cirne, W., Brasileiro, F. V., Andrade, N., Costa, L., Andrade, A., Novaes, R., Mowbray, M. (2006). Labs of the World, Unite!!! J. Grid Comput. 4(3): 225-246.

Défago, X., Schiper, A., Úrban, P. Total order broadcast and multicast algorithms: Taxonomy and survey. ACM Computing Surveys 36(4): 372-421 (2004).

Défago, X., Urbán, P., Hayashibara, N., Katayama, T. (2005). Definition and Specification of Accrual Failure Detectors. In Poc. DSN'2005.

de Sá, A. S. and Macêdo, R. J. A. (2009). Uma Proposta de Detector de Defeitos Autonômico Usando Engenhria de Controle. In Workshop de Testes e Tolerância a Falhas (WTF'2009), João Pessoa, Brasil.

Fischer, M., Lynch, N., Peterson, M (1985). Impossibility of distributed consensus with one faulty process. In Journal of ACM, Volume 32, Issue 2, pages 374-382.

Gartner, Felix C. (1999) Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. In ACM Computing Surveys, Vol. 31 No. 1, pages 1 – 26, New York, NY, USA.

Goldchleger, A., Kon, F., Goldman, A., Finger, M., Bezerra, G.C. (2004). InteGrade: object-oriented Grid middleware leveraging the idle computing power of desktop machines . Concurrency - Practice and Experience 16(5): 449-459.

Guerraoui, R., Rodrigues, L. (2006). Introduction to Reliable Distributed Programming, Springer-Verlag.

Hayashibara, N., Défago, X., Yared, R. and Katayama, T. (2004). The fi accrual failure detector. In 23º, IEEE International Symposium on Reliable Distributed Systems, pages 66–78. Florianópolis, Brazil,

Jacobson, V. (1988). Congestion avoidance and control. In SIGCOMM '88: Symposium proceedings on Communications architectures and protocols, pages 314–329, New York, NY, USA.

Satzger, B., Pietzowski, A., Trumler, W., Ungere, T. (2007). A New Adaptive Accrual Failure Detector for Dependable Distributed Systems. In ACM Symposium on Applied Computing, pages 551-555, Seoul, Korea.

Schneider, F. B., Gries, D., Schlichting, R. D (1984). Fault-Tolerant Broadcasts. Sci. Comput. Program. 4(1): 1-15

Sens, P., Arantes, L., Bouillaguet, M., Simon, V., Greve. F. (2008). An Unreliable Failure Detector for Unknown and Mobile Networks. 12th International Conference on Principles of Distributed Systems, pages 555-559, Luxor, Egypt.

Sotoma, I. and Madeira, E. R. M. (2001). ADAPTATION – Algoritms to ADAPTive FAulT MonItOriNg and their implementation on CORBA. In 3rd International Symposium os Distributed Objects and Applications, Rome, Italy.

Tanscheit, R. (2003) Sistemas Fuzzy. Apostila de Mini Curso. Bauru, SP: VI Simpósio Brasileiro de Automação Inteligente, 2003.

Xiong, N., Défago, X. (2007). ED FD: Improving the Phi Accrual Failure Detector. Research Report IS-RR-2007-007, Japan Advanced Institute of Science and Technology, Ishikawa, Japan, April 2007.

Zadeh, L.A. (1965). Fuzzy Sets. Information and Control, V. 8: 338-353.