Uma Proposta de Detector de Defeitos Autonômico Usando Engenharia de Controle

Alirio Santos de Sá, Raimundo José de Araújo Macêdo

[1] Laboratório de Sistemas Distribuídos (LaSiD)
Departamento de Ciência da Computação
Instituto de Matemática, Universidade Federal da Bahia
CEP 40170-110 – Salvador, Bahia, Brasil

{aliriosa,macedo}@ufba.br

Resumo. *A escolha do período de monitoramento, na construção de detectores de defeitos, é a chave para uma detecção mais rápida e com um menor consumo de recursos. Todavia, essa escolha não é uma tarefa trivial quando são considerados ambientes com atrasos computacionais desconhecidos e que podem mudar dinamicamente com o tempo. Assim, o presente artigo apresenta e avalia, usando métricas de Qualidade de Serviço (QoS) de detecção, uma estratégia, baseada em teoria de controle, para sintonia automática do período de monitoramento. Tal estratégia visa adequar o consumo de recursos do detector à quantidade de recursos disponíveis no ambiente, atendendo aos requisitos de QoS de detecção.*

Abstract. *Choosing a proper monitoring period for the implementation of failure detectors is the key to faster detection with lower consumption of resources. However, this choice is not a simple task when computing environments present unknown computational delays that can vary dynamically over time. This paper faces such a challenge by presenting and evaluating a strategy based on control theory for the automatic tuning of the monitoring period. Such a strategy aims also to adapt the resource consumption of the detector to the amount of available resources, according to some detection QoS requirements.*

1. Introdução

Detectores de defeitos são blocos básicos na construção de mecanismos de tolerância a falhas[1]: seja para ativar procedimentos de recuperação, seja para permitir a reconfiguração do sistema.

Em sistemas distribuídos, o monitoramento remoto do defeito em componentes do sistema deve ser feito através de troca de mensagens. Quando os atrasos de comunicação ou computação variam de forma não determinística, a detecção de defeitos não é precisa e o detector pode realizar uma detecção equivocada: tomando como falhos processos que efetivamente nunca falharam ou deixando de apontar a falha de processos que efetivamente tenham falhado. Nesses cenários, os detectores não são confiáveis [Chandra and Toueg 1996]. Para melhorar a confiabilidade e a velocidade dos detectores na detecção das falhas é preciso que os mesmos possam se adaptar a variabilidade dos atrasos impostos pelo ambiente [Macêdo and Lima 2004].

[1] adotamos a nomenclatura *falha-erro-defeito* para *fault-error-failure*

Além disso, para minimizar o custo com troca de mensagens ou processamento e permitir que os detectores possuam a confiabilidade e a velocidade necessárias, é preciso que os projetistas de sistemas tenham conhecimento de como os algoritmos de adaptação se comportam dentro do ambiente no qual os mesmos serão implementados. Ou seja, conhecer como os algoritmos de adaptação funcionam é importante para realizar a sintonia dos parâmetros de tais algoritmos e obter o nível de QoS de detecção desejado.

Todavia, quando o ambiente muda com o tempo (e.g. mudança na largura de banda disponível, falhas de componentes, mobilidade etc.), ou quando os requisitos das aplicações (e.g. tempo de resposta, número de transações por segundo etc.) mudam, é necessário que uma nova sintonia dos parâmetros de detecção seja realizada, tanto para garantir o nível de qualidade de serviço requerido, quanto para manter o consumo de recursos dentro dos limites esperados. A depender do quão rápido o ambiente e os requisitos mudam, realizar a sintonia dos mecanismos de tolerância a falhas, e em particular dos detectores defeitos, pode não ser uma tarefa simples e em muitos casos impossível de serem realizadas com intervenção humana. Assim, é necessário que os mecanismos de detecção de defeitos sejam dotados de características autonômicas [Huebscher and McCann 2008] e possam realizar a auto-sintonia de seus parâmetros de modo a acomodar as mudanças do ambiente ou dos requisitos das aplicações de forma automática. Entretanto, a maioria dos parâmetros associados à sintonia dos detectores adaptativos está diretamente relacionada aos algoritmos de adaptação. Ou seja, uma proposta de sintonia automática baseada em tais parâmetros será estritamente dependente do algoritmo de adaptação usado.

A maioria das abordagens de detecção adaptativa usam mensagens de monitoramento de forma periódica [Falai and Bondavalli 2005, Nunes and Jansch-Pôrto 2004]. Esse período dita de quanto em quanto tempo os componentes serão monitorados. Desse modo, o período de monitoramento é um parâmetro que pode ser usada no processo de auto-sintonia e que garante um ajuste com certo grau de independência do algoritmo de adaptação usado pelo detector. Além disso, esse período está diretamente relacionado à velocidade, à precisão e ao consumo de recursos de comunicação e computação do detector adaptativo. Quanto menor for o período de monitoramento, maior será o consumo de recursos de computação e comunicação, porém, menor também será o tempo de detecção e mais rápido o algoritmo de adaptação irá aprender sobre o estado atual do ambiente. Todavia, um período muito pequeno pode aumentar a variabilidade dos atrasos de computação e comunicação, implicando em um aumento no tempo de detecção e na diminuição da precisão do detector.

Muitas estratégias têm sido usadas para implementar autonomia em sistemas computacionais [Huebscher and McCann 2008]. Dentre essas estratégias, a teoria de controle tem sido bastante utilizada [Hellerstein et al. 2004]. Nessa abordagem, um arcabouço matemático para a construção de mecanismos de controle de sistemas dinâmicos é usado para modelar e analisar o sistema como uma caixa-preta, através apenas das relações de entrada e saída [Ogata 1995]. Esse arcabouço matemático permite que diversas características do sistema sejam observadas e definidas ainda em tempo de projeto (e.g. estabilidade, precisão, convergência, etc.).

1.1. Contribuições

Nesse contexto, o presente trabalho apresenta e avalia uma proposta de detecção autonômica de defeitos, implementada usando teoria de controle e capaz de realizar a sintonia automática de parâmetros de detectores adaptativos de defeitos com o intuito de obter a qualidade de serviço desejada pela aplicação. Mais especificamente, o presente artigo:

- apresenta uma estratégia inovadora para o ajuste do período de monitoramento de detectores de defeitos, usando uma estratégia de controle Propocional-Integral (PI), considerando cenários nos quais o comportamento do ambiente são desconhecidos;
- permite a utilização de diferentes abordagens de detecção adaptativa sem considerar detalhes da implementação das mesmas.

Cabe ressaltar que o ponto chave da contribuição é obter uma proposta preliminar de um detector de defeitos efetivamente autonômico, que possa se ajustar a novos requisitos de QoS e do ambiente computacional, diferentemente de trabalhos anteriores que, em sua maioria, visam a adaptação dinâmica de *timeouts* de detecção.

1.2. Trabalhos Correlatos

Diversos estudos têm posposto e avaliado diferentes implementações de detectores. As propostas de [Macêdo 2000], [Bertier et al. 2003], [Müller 2004], [Nunes and Jansch-Pôrto 2004], [Sampaio and Brasileiro 2004], [Macêdo and Lima 2004] e [Falai and Bondavalli 2005], por exemplo, focam apenas na predição dos atrasos relacionados ao tempo de viagem das mensagens e nenhuma delas, diferente da proposta apresentada neste artigo, considera o ajuste do período de monitoramento de modo a adequar o consumo de recursos do detector face a mudanças no ambiente. Além disso, a maioria dessas abordagens apenas usam as métricas de QoS para avaliação do desempenho não determinando como o ajuste do detector pode ser concebido a partir das mesmas. [Chen et al. 2002] propõem e utilizam métricas de QoS no processo de sintonia do detector. Entretanto, esse trabalho considera um ambiente com um comportamento probabilístico conhecido e não realiza o ajuste do período de monitoramento durante a execução do detector. A proposta aqui apresentada, por outro lado, não pressupõe qualquer conhecimento a priori a cerca do ambiente. [Xiong et al. 2006] apresenta um detector baseado em teoria de controle para o ajuste do período de monitoramento do detector. Todavia, esse trabalho considera que as características do ambiente são conhecidas a priori e não considera métricas de qualidade de serviço na sintonia do detector. [Mills et al. 2004] propõe um detector autonômico, no qual o período de monitoramento é ajustado para minimizar o impacto do detector no consumo de recursos do ambiente. Contudo, essa abordagem não utiliza uma estratégia baseada em engenharia de controle, considera que características do ambientes são conhecidas e não avalia o desempenho do detector usando métricas de QoS. Além disso, não considera uma implementação de detector autonômico que estenda as facilidades providas por um detector adaptativo.

1.3. Estrutura do Artigo

Este artigo está organizado da seguinte forma. Na seção 2 são discutidos aspectos relacionados à adaptabilidade e QoS de detecção. Na seção 3 é apresenta a proposta de detecção autonômica. Na seção 4, avalia-se o desempenho do detector autonômico, comparando-o com detectores adaptativos baseados em período de monitoramento fixo. Por fim, na seção 5, são apresentadas as considerações finais e propostas para trabalhos futuros.

2. Detectão de Defeitos: Adaptabilidade e Qualidade de Serviço

Em um ambiente distribuído, a implementação de detectores de defeitos considera dois modelos básicos para o monitoramento remoto do estado de componentes do sistema: *Pull* e *Push* [Felber 1998]. No modelo *Pull*, figura 1(a), uma vez recebida uma mensagem de *"Are you Alive?"* do módulo monitor do detector de defeitos, o componente monitorado deve responder com seu atual estado (mensagem de *"I am alive!"* ou *heartbeat*). A cada mensagem de monitoramento enviada, o monitor deve estimar o intervalo de tempo necessário (*timeout*) para a chegada da mensagem de resposta oriunda do componente monitorado. Caso a mensagem não chegue dentro do intervalo esperado, o detector passa a suspeitar da falha do componente. No modelo *Push*, figura 1(b), o componente monitorado espontaneamente envia seu estado atual. Baseado no intervalo entre chegada das mensagens, o componente monitor deve estimar o instante de chegada da próxima mensagem de detecção. Caso a mensagem não chegue dentro do intervalo esperado, o detector suspeita da falha do componente.

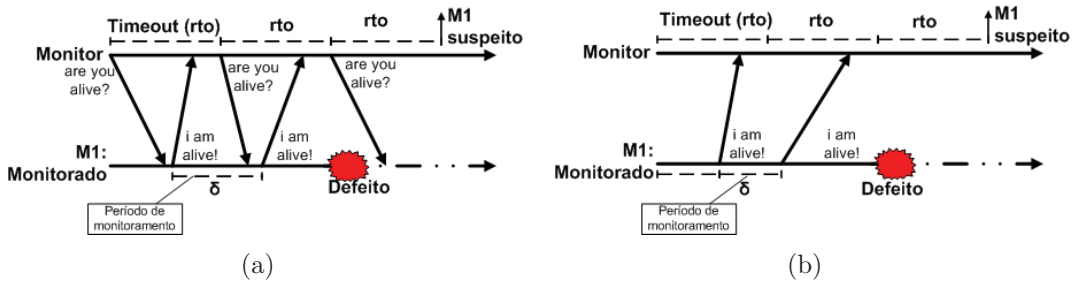

Figure 1. Modelos de Monitoramento:(a) Pull; (b) Push

Usando o modelo *Pull*, o detector consegue não apenas ter uma estimativa mais eficiente dos atrasos de comunicação, mas também pode controlar de forma mais eficiente o período de monitoramento. No modelo *Push*, entretanto, o número de mensagens de detecção trocadas entre os módulos do detector é menor, o que implica em um menor consumo de recursos do canal de comunicação.

2.1. Adaptabilidade na Detecção de Defeitos

A discussão sobre adaptabilidade desta seção foca abordagens que utilizem o modelo de monitoramento *Pull*, entretanto, as questões aqui apontadas podem facilmente ser estendidas para o modelo *Push* (ver [de Sá and Macêdo 2005]).

A fim de facilitar a discussão a seguir, considera-se a existência de apenas dois componentes p e q, em que o componente q possui um módulo monitor do detector de

defeitos embutido e monitora falhas do componente p. A cada δ unidades de tempo, q questiona o estado de p, enviando uma mensagem, seqüencialmente assinalada e denotada por *aya* ("are you alive?"). O componente p deve informar que está funcionando corretamente enviando uma mensagem, dita *heartbeat* (*hb*). Para cada mensagem enviada, q deve calcular o intervalo de tempo (*rto*) necessário para que um *heartbeat*, oriundo de p, retorne. Se *hb* não chega dentro de *rto* unidades de tempo, q coloca p em sua lista de suspeitos. Caso q receba um *hb* com um número seqüencial igual ou superior ao do *heartbeat* esperado, q remove p de sua lista de suspeitos.

O componente q envia mensagens *aya* em instantes denotados por σ, dessa forma: aya_k é enviada no instante σ_k, aya_{k+1} em σ_{k+1}, aya_{k+2} em σ_{k+2} e assim sucessivamente. Para quaisquer dois instantes consecutivos σ_k e σ_{k+1}, tem-se: $\sigma_{k+1} - \sigma_k = \delta$. Os instantes de chegada das mensagens *hb* são denotados por α, ou seja: hb_k chega em α_k, hb_{k+1} em α_{k+1}, hb_{k+2} em α_{k+2} e assim por diante. Desta forma, em um ambiente no qual não há variação no atraso nem perda de mensagens no subsistema de comunicação, os *heartbeats* chegam em q espaçados entre de si de exatamente δ unidades de tempo. Assim, sendo rtt o tempo necessário para a transmissão de um *aya* de q para p e retorno de um *hb*, tem-se: $\alpha_{k+1} = \sigma_{k+1} + rtt$. Com isso, o intervalo entre as chegadas das mensagens hb_k e hb_{k+1} será $\Delta\alpha_{k+1} = \alpha_{k+1} - \alpha_k$ (ou $\Delta\alpha_{k+1} = \delta$). Nesses cenários, cada *heartbeat* carrega uma informação, do estado de p, envelhecida de aproximadamente $\frac{rtt}{2}$ unidades de tempo, isto considerando que os tempos de ida de um *aya* e retorno de um *hb* são iguais. Se p envia hb_k e falha em seguida, q receberá hb_k e só suspeitará da falha do componente p quando não receber hb_{k+1}. Para o componente q, entretanto, p pode ter falhado em qualquer instante de tempo entre $\sigma_k + \frac{rtt}{2}$ e $\sigma_{k+1} + \frac{rtt}{2}$. Entretanto, q só terá ciência da falha de p em $\sigma_{k+1} + rtt$ (assumindo $rto \approx rtt$). Desse modo, o menor intervalo de tempo no qual q pode começar a suspeitar da falha de p com segurança não pode ser menor que $\delta + \frac{rtt}{2}$. Portanto, o tempo mínimo para detecção segura de uma falha é $min(t_d) = \delta + \frac{rtt}{2}$.

Se variações no atraso são consideradas, cada mensagem hb_k terá um rtt_k associado. Como $\alpha_k = \sigma_k + rtt_k$, observa-se que $\alpha_{k+1} - \alpha_k = \delta + (rtt_{k+1} - rtt_k)$, ou $\Delta\alpha_{k+1} = \delta + \Delta rtt_{k+1}$. Quando os tempos de viagem das mensagens não são conhecidos e não há relógios sincronizados, as estimativas do detector de defeitos não são precisas e não é possível estimar o tempo de detecção com exatidão. Sendo $\hat{rtt}_k$ a estimativa realizada pelo detector para o intervalo de tempo necessário para a transmissão de um aya_k e recebimento de um *heartbeat* hb_k, quanto mais próximo $\hat{rtt}_k$ estiver de rtt_k menor será o tempo de detecção. A relação $\hat{rtt}_k \geq rtt_k$ deve ser satisfeita para que o detector evite falsas suspeitas. Assim, a relação entre $\hat{rtt}$ e rtt é uma indicação do desempenho do detector em termos de tempo de detecção. Uma vez que variações extras no atraso podem provocar subestimativas, utiliza-se uma margem de segurança (ρ) que compense variações não previstas no atraso da rede [Chen et al. 2002]. Desta forma, o intervalo estimado de tempo *rto* para a chegada do *heartbeat* hb_{k+1} é definido por: $rto_{k+1} = \hat{rtt}_{k+1} + \rho_{k+1}$. Portanto, a adaptabilidade do detector consiste em ajustar $\hat{rtt}$ e ρ [Nunes and Jansch-Pôrto 2004]. Assim, uma vez recebido hb_k em α_k, quanto mais precisa a estimativa de $\hat{rtt}_{k+1}$ e ρ_{k+1} mais próximo rto_{k+1} estará de rtt_{k+1}.

2.1.1. A Abordagem de Adaptação de Jacobson [Jacobson 1988]

Dentre as diversas abordagens de adaptação para detectores de defeitos, este artigo destaca o trabalho de [Jacobson 1988]. Nesse trabalho, realiza-se a predição usando uma suavização exponencial do erro entre os valores medido e estimado. Dessa forma, assumindo $err_k = rtt_k - \hat{rtt}_k$, tem-se: $\hat{rtt}_{k+1} = \hat{rtt}_k + \gamma \cdot err_k$, em que γ, $\hat{rtt}$ e rtt representam, respectivamente, a confiança no erro calculado (err_k), a estimativa e o valor efetivamente medido para o atraso de *ida-e-volta*. Por fim, Jacobson calcula uma margem de segurança da seguinte forma: $\rho_{k+1} = \rho_k + \gamma \cdot (|err_k| - \rho_k)$. Assim, é possível obter o *timeout* por: $rto_{k+1} = \beta \cdot \hat{rtt}_{k+1} + \phi \cdot \rho_{k+1}$, em que ϕ e β ponderam a influência de $\hat{rtt}$ e ρ na estimativa do *timeout*.

2.2. Qualidade de Serviço na Detecção de Defeitos

A detecção de defeitos em ambiente distribuído está diretamente ligada à velocidade dos processos e aos atrasos no canal de comunicação. Sendo assim, se existem variações não determinísticas no tempo de processamento ou no tempo de transmissão das mensagens, o detector pode cometer erros em sua detecção. Por conta disso, tais detectores são ditos não confiáveis, uma vez que podem suspeitar de componentes que efetivamente nunca falharam (ditos componentes corretos) ou ainda não suspeitar componentes que efetivamente falharam. Os autores de [Chandra and Toueg 1996] propuseram uma classificação para os detectores de defeitos, a qual varia de acordo com duas propriedades básicas: *Completeness* – referindo-se a capacidade dos detectores de suspeitar, em algum momento futuro, de componentes que efetivamente falharam; e *Accuracy* – relacionada com a capacidade do detector em evitar falsas suspeitas. Essa classificação, entretanto, é baseada em comportamentos temporais futuros não quantificáveis, sendo, portanto, não apropriada para algumas aplicações que necessitem de garantias temporais de detecção mais fortes. Tais aplicações precisam de métricas que possam mensurar a qualidade de serviço prestada pelos mecanismos de detecção de defeitos [Hermant and Lann 2002]. Em [Chen et al. 2002], são propostas métricas probabilísticas de *QoS*, (*Quality of Service*) para medir a capacidade do detector em prevenir falsas suspeitas (*accuracy*) e a rapidez com a qual o mesmo detecta a falha dos componentes (*speed*). As principais métricas são: (i) *Tempo de detecção* (*Detection Time*, *td*) – intervalo de tempo entre o instante em que o componente monitorado falhou e o instante no qual detector passou a suspeitar definitivamente da falha do mesmo; (ii) *Intervalo entre falsas suspeitas consecutivas* (*Mistake Recurrence Time*, *tr*); (iii) *Tempo para correção de uma falsa suspeita* (*Mistake Duration*, *tm*).

3. Uma Proposta para Detecção Autonômica

A idéia principal da proposta é prover ao detector adaptativo a capacidade de autosintonia, abstraindo, contudo, detalhes de implementação do algoritmo utilizado e do ambiente de execução do mesmo . Desta forma, diferentes abordagem de adaptação podem ser utilizadas e diferentes ambientes podem ser considerados. Para tanto, a proposta considera a implementação de um mecanismo de gerenciamento (*gestor autonômico*[2] ou *controlador*[3]), o qual deve perceber o estado atual do detec-

[2]No jargão da área de computação autonômica [Huebscher and McCann 2008].
[3]No jargão da área de engenharia de controle [Hellerstein et al. 2004].

tor adaptativo (*elemento gerenciado*[2] ou *planta*[3]) a partir do histórico das mensagens do mesmo e ajustar o período de monitoramento de modo a atender os requisitos de alto nível considerados (QoS de detecção e consumo de recursos, ver figura 2).

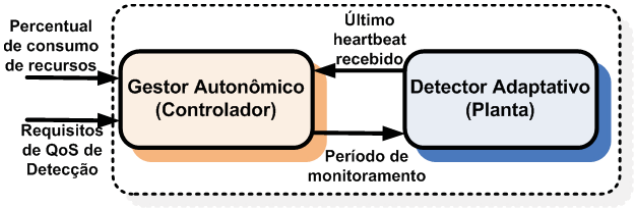

Figure 2. Detector Autonômico

3.1. Considerações de Projeto

Para a proposta, o projeto do detector autonômico considera como objetivo a redução do tempo de detecção, usando para tanto um percentual da quantidade de recursos disponíveis. A quantidade de recursos disponíveis varia de acordo com a carga no ambiente (e possíveis falhas). Sendo assim, quanto maior a carga (ou o número de falhas), menos recursos estarão disponíveis. Da mesma forma, quanto menor a carga, maior será a quantidade de recursos disponíveis. Vale observar que o problema não é trivial, uma vez que o projeto considera um modelo caixa-preta, no qual detalhes a respeito do ambiente não estão disponíveis (e.g. largura de banda, número de processos, topologia da rede, probabilidade de perda de mensagens, etc.). Nesta proposta, o consumo de recursos é percebido apenas através dos atrasos de ida-e-volta das mensagens de monitoramento do detector.

Para obter um ajuste mais eficiente do período de monitoramento, a proposta considera como planta um detector adaptativo baseado no modelo *Pull*. Por fim, para manter os requisitos de QoS, a quantidade mínima de recursos é limitada pelo tempo máximo de detecção. Entretanto, essa quantidade mínima de recursos não é garantida, pois, no modelo, não são consideradas reservas de recursos para o detector.

Para atender aos principais objetivos do projeto, quatro aspectos básicos devem ser considerados: [a] formulação do problema de controle; [b] definição do modelo matemático da planta; [c] determinação da lógica do laço de controle; e [d] projeto e sintonia do controlador. Esses aspectos são discutidos a seguir.

3.1.1. Formulação do Problema de Controle

A formulação do problema de controle diz respeito à definição do objetivo do controle e das variáveis associadas ao problema. Conforme discutido anteriormente, a variável de entrada da planta (detector adaptativo), dita u_k, é representada pelo período de monitoramento (δ), ou $u_k = \delta_k$. O objetivo final da proposta é permitir um tempo de detecção mínimo com mínimo impacto possível nos atrasos computacionais do ambiente. Esse tempo de detecção, conforme discutido na seção 2.1, dependerá efetivamente do atraso de ida-e-volta de uma mensagem de monitoramento (ou da estimativa realizada do mesmo) e da magnitude do período de monitoramento. Assim, é preciso estabelecer estimativas para o tempo de detecção, de modo

que se possa verificar como o mesmo se comporta em função da variação da carga computacional no ambiente. Assim, de forma bastante conservadora, uma vez conhecidos os atrasos de ida-e-volta, pode-se calcular o suposto tempo de detecção (*sptd*, *supposed detection time*) por: $sptd_k = \alpha_k - (\sigma_{k-1} + \frac{rtt_{k-1}}{2})$. Da mesma forma, o limite inferior para o suposto tempo de detecção (*bstd*, *best-case of supposed detection time*) pode ser calculado por $bstd_k = \alpha_k - \left(\sigma_{k-1} + \frac{1}{2} \cdot rttmin_k\right)$, ou aproximadamente $bstd_k = \delta_k + \frac{1}{2} \cdot rttmin_k$. Em que $rttmin_n = \{rtt_i | rtt_i \leq rtt_j; \forall i, j \leq n\}$. Deseja-se que *sptd* esteja o mais próximo possível de *bstd*, desse modo, pode-se adotar como variável de saída (y) a distância entre *sptd* e *bstd*, assim $y_k = sptd_k - bstd_k$. Se a carga computacional no ambiente é baixa, então $y_k \approx 0$ (o suposto tempo de detecção está próximo do seu limite mínimo). Por outro lado, se a carga no ambiente se eleva, a variável de saída terá um valor diferente de zero, sendo este valor máximo quando $rtt_n = rttmax_n$, com $rttmax_n = \{rtt_i | rtt_i \geq rtt_j; \forall i, j \leq n\}$. Se $rttmax \gg rttmin$, a variável de saída (y) pode não ser sensível a variações dos valores de entrada (u). Desse modo, a saída precisa ser normalizada dentro da faixa entre 0 e 1. Por conta disso, a variável de saída pode ser reescrita como $y_k = \frac{sptd_k - bstd_k}{lnorm_k}$, com:

$$lnorm_k = \begin{cases} 1, & se\ rttmin_k = rttmax_k \\ rttmax_k - rttmin_k, & se\ rttmin_k \neq rttmax_k \end{cases}$$

Por conta das variações na carga computacional do ambiente, a variável *sptd* pode possuir uma variabilidade bastante significativa. Essa variabilidade pode fazer com que a saída da planta oscile de forma, além de induzir oscilações indesejadas nos valores sugeridos para o período de monitoramento. Por conta disso, é interessante reduzir essa variabilidade através de uma suavização do efeito das variações de *sptd* na variável de saída. Portanto, e finalmente, pode-se representar a variável de saída da planta por:

$$y_k = \frac{\frac{sptd_k + sptd_{k-1}}{2} - bstd_k}{lnorm_k} \tag{1}$$

O diagrama da figura 3(a) representa o modelo do sistema em malha aberta. Conforme pode ser visto na figura, os valores de *sptd* e *bstd* são obtidos a partir do histórico das mensagens de monitoramento do detector de defeitos adaptativo. Para tanto, cada *heartbeat* deve conter o um número seqüencial, o timeout atribuído ao mesmo e os instantes de envio e recebimento. Na figura 3(a), o transdutor é responsável em transformar o sinal de *heartbeat* em sinais de saída (y).

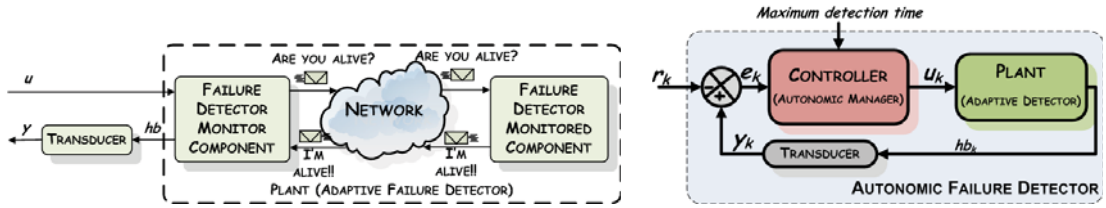

Figure 3. Modelos do Sistema:(a) Malha aberta; (b) Malha fechada

3.1.2. Lógica do laço de controle

Uma vez definidas as variáveis e o relacionamento das mesmas no problema de controle, o próximo passo é descrever como o controlador deve atuar no laço (ou malha) de controle. No problema de controle proposto, o objetivo da ação de controle é otimizar $u_k = \delta_k$ (período de monitoramento). Assim, na medida em que a carga computacional no ambiente cresce, u_k deve ser incrementado de modo a reduzir o impacto da planta (detector adaptativo) sobre os atrasos no ambiente. Da mesma forma, quando a carga no ambiente diminui, u_k deve ser decrementado para garantir o menor tempo de detecção possível ($sptd_k = bstd_k$). Um diagrama do laço de controle em malha fechada é apresenta na figura 3(b). Nessa figura, r_k (o *set-point*) representa o percentual da carga computacional disponível que pode ser usada pelo detector adaptativo. Para tanto, y_k é comparado com r_k e a diferença $e_k = y_k - r_k$ é enviada para o controlador. Quando o erro é negativo ($e_k < 0$), indica que existem recursos disponíveis, logo o controlador pode reduzir u_k. Se $e_k > 0$, indica uma redução na quantidade de recursos disponíveis, logo u_k precisa ser incrementado.

3.1.3. Definição Modelo da Planta

A partir da lógica do laço de controle apresentado na subseção anterior, foi construído um modelo ARX (*Auto-Regressive eXogenous*[4]) sobre dados coletados durante experimentos que consideram diferentes condições de carga no ambiente, de modo a permitir uma identificação do modelo da planta a partir das relações de entrada e saída[4]. Apesar de a planta possuir um comportamento não linear, optou-se pela utilização de um modelo ARX de primeira ordem: $y_{k+1} = c \cdot y_k + (1 - c) \cdot u_k$. Em que c pondera a influência da carga atual (y_k) e do período de monitoramento do detector na próxima carga computacional (y_{k+1}) que será gerada.

De posso do modelo ARX apresentado, é possível derivar a função de transferência da planta, usando a transformada Z [Ogata 1995]: $P(z) = \frac{Y(z)}{U(z)} = \frac{1-c}{z-c}$. A notação em Z é bastante utilizada porque facilita a análise da características do sistema, permitindo a construção de um controlador que atenda aos requisitos de estabilidade, robustez e precisão (ver [Ogata 1995]).

3.1.4. Projeto do Controlador

Controladores *PID* (*Proporcional-Integral-Derivativo*, ver [Ang et al. 2005]) e suas variações (P, PD e PI) têm encontrado grande aceitação na automação de processos industriais e nas mais diversas aplicações das mais diversas áreas [5]. Assim, neste trabalho, considera-se um controlador PI (Proporcional-Integral), cujo modelo ARX e a função de transferência em Z são[4]: $u_k = u_{k-1} + (K_p + K_i) \cdot e_k - K_p \cdot e_{k-1}$ e $C(z) = K_p + K_i \cdot \left(\frac{z}{z-1}\right)$. Em que u_k, e_k, K_p e K_i representam, respectivamente, a saída do controlador, o desvio entre a resposta desejada e a obtida após a ação de controle e os ganhos proporcional e integral do controlador.

[4]Maiores detalhes podem ser encontrados em [Hellerstein et al. 2004].

[5]e.g. [Fengyuan and Chuang 2003], [Majumdar et al. 2004], [Ang et al. 2005], entre outros.

O processo de sintonia do controlador (determinação dos ganhos K_p e K_i) deve considerar critérios objetivos para o desempenho do controle. Durante o projeto de controladores para sistemas computacionais, foca-se em quatro propriedades para o laço de controle [Hellerstein et al. 2004]: Estabilidade (Stability), Precisão (Accuracy), Tempo de convergência (Settling times) e Variação máxima (Overshoot). Um sistema é dito estável se para uma entrada limitada produz uma saída limitada (ver [Simon 2002]). Por sua vez, um sistema é dito preciso se consegue minimizar o erro entre a resposta desejada e a obtida (Steady-state error). O tempo de convergência (K_s) representa o tempo que o sistema leva para atingir o estado desejado. A variação máxima (M_p) é a diferença máxima entre y_k e r_k.

A estabilidade do sistema é determinada pela localização dos pólos do laço de controle [$G(z) = \frac{P(z)\cdot C(z)}{1+P(z)\cdot C(z)}$]. Os pólos do sistema são a solução para a equação característica $1+P(z)\cdot C(z) = 0$. O sistema será estável se todos os pólos do sistema tiverem magnitudes menores que 1 [Simon 2002].

O tempo de estabilização (K_s) e o máximo sobre sinal (M_p) são usados para determinar os parâmetros do controlador PI. Para tanto, use-se o procedimento de projeto baseado na localização dos pólos [Hellerstein et al. 2004]. Assim, assumindo que os pólos em malha fechada são conjugados complexos ($r^{\pm j\theta}$), é possível calcular a localização dos pólos, usando K_s e M_p especificados: $r \approx \exp(\frac{-4}{k_s})$ e $\theta \approx \pi \cdot \frac{\log r}{\log M_p}$. Desta forma, é possível encontrar determinar: $K_p = \frac{c-r^2}{1-c}$ e $K_i = \frac{r^2-2\cdot r\cdot\cos\theta+1}{1-c}$.

Consideração gerais sobre o controlador. Diferentes das abordagens de controle de processos industriais, as ações de controle, no contexto do detector autonômico, não podem assumir qualquer valor (e.g. não são permitidos valores negativos para o período de monitoramento). Desse modo é preciso definir uma faixa de valores possíveis para o período de monitoramento. Essa faixa de valores está associada com o tempo máximo de detecção ($tdmax$) e com $rttmin$. Assim, o período de monitoramento deve está contido no intervalo $[\frac{rttmin}{2}, tdmax - \frac{rttmin}{2}]$. Vale salientar que $tdmax$ é um parâmetro fornecido pelo usuário e diz respeito ao requisito de QoS de detecção requerido pela aplicação. Além disso, estratégias tradicionais de controle, se baseiam na coleta periódica de amostras da planta, o que exige um conhecimento da dinâmica de tal planta. No caso da proposta, é difícil estabelecer um período de amostragem que seja conveniente a execução do controle, uma vez que a magnitude dos atrasos do ambiente são desconhecidas e podem mudar com o tempo. Assim, no projeto da proposta, optou-se por realizar a amostragem da planta a cada evento associado ao instante de chegada de uma mensagem de *heartbeat* no módulo monitor do detector adaptativo. Desse modo, ao invés de fazer referência ao tempo real, um ciclo de controle considera rodadas de execução do detector. Essas rodadas estão associadas ao ciclo de monitoramento, o qual é definido pelo intervalo entre coletas de informação do componente remoto. Esse intervalo variar dependendo da magnitude de rtt e δ. Neste caso, os valores indicados para K_s, por exemplo, estarão expressos em ciclos de monitoramento (ou rodadas).

4. Avaliação de Desempenho

4.1. Descrição do Ambiente Simulação

Para avaliação de desempenho, considera-se um ambiente simulado usando o *Simulink*[The Mathworks 2006] do *Matlab*[The Mathworks 2002], com o auxilio do

pacote *TrueTime 1.5*[Henriksson and Cervin 2003].

O sistema simulado considera três computadores, identificados por c_1, c_2 e c_3, interligados por uma rede *Switched Ethernet*. Em c_1 executa o módulo monitor do detector de defeitos, o qual monitora defeitos do computador c_2. O computador c_3 é utilizado para gerar rajadas aleatórias de trafego na rede. Para tanto, c_3 utiliza o procedimento listado no algoritmo 1. Nesse algoritmo, o procedimento em execução em c_3 é ativado periodicamente a cada $itp = \frac{msize}{nwrate}$ unidades de tempo, em que $msize$ e $nwrate$ representam, respectivamente, o tamanho das mensagens em bits e a taxa nominal de transferência da rede em bits por segundo (bps). A variável $bw \in (0, 1)$ é uma variável aleatória que segue uma distribuição normal e indica o percentual de largura de banda na rede que será ocupada pelo procedimento $interf$. A cada $10ms$, o limite de ocupação da largura de banda ($nwload$) é incrementado em 10%. Se $bw < nwload$ então uma mensagem vazia é enviada ao computador c_1.

Algoritmo 1. Procedimento para geração de carga

```
1  procedure interf{
2   for each time interval itp{
3    bw = rand(1); t = currentTime(); nwload = mod(floor(t · 10^-6), 10 · 10^-3) * 0.1;
4    if (bw < nwload) send null to c1;
5   }}
```

O *Switch Ethernet* possui uma taxa nominal de transferência de $10Mbps$ e todas as mensagens transmitidas através do mesmo são representadas por quadros de tamanho fixo e iguais a 1518 bits. Além disso, *Switch Ethernet* possui um $buffer$ com capacidade para 52 mensagens e em caso de congestionamento, *buffer overflow*, acontece o descarte de mensagens.

Consideram-se, ainda, seis detectores usando a estratégia de [Jacobson 1988], cada uma com uma configuração diferente. Cinco das configurações utilizam períodos de monitoramento fixos e equivalentes a 100, 200, 300, 500 e 800 microssegundos, respectivamente. A sexta configuração representa o detector autonômico, no qual o período de monitoramento é ajustado de forma automática. A estratégia de Jacobson possui os mesmos parâmetros de predição para todos os seis detectores: $\gamma = 0,1$; $\beta = 1,0$; e $\phi = 2,0$ (conforme originalmente proposto em [Jacobson 1988]). O detector autonômico é configurado da seguinte forma: $c = 0.9$; set-point $r = 10\%$ dos recursos computacionais disponíveis; $K_s = 10$ rodadas; $M_p = 20\%$; e $tdmax = 10ms$. A partir desses parâmetros são especificados os ganhos do controlador $K_p = 4,51$ e $K_i = 4,97$. Para avaliação de desempenho são consideradas as métricas de QoS de detecção (td, tm e tr) e o número de falsas suspeitas cometidas por cada detector. Por fim, todas as simulações foram executadas considerando um tempo simulado de 40 segundos, o que representa, no mínimo, aproximadamente 10^5 mensagens de monitoramento para cada um dos detectores de defeitos.

4.2. Avaliação dos Resultados da Simulação

A figura 4 apresenta o resultado da avaliação de desempenho, em função dos valores médios das métricas de QoS de detecção e do número de falsas suspeitas. Nessa figura, a primeira coluna representa o percentual de ocupação da rede pelo processo $interf$. As colunas de 2 a 6 representam os detectores adaptativos com período de monitoramento fixo. A coluna 7 representa a proposta de detecção autonômica.

Comparado com os demais, o detector autonômico (AFD) obteve melhor desempenho em termos do tempo médio de detecção e da duração média da falsa suspeita. Vale salientar que, o detector adaptativo com menor período de monitoramento ($100\mu s$), foi o que obteve desempenho mais próximo do detector autonômico nessas duas métricas. Contudo, para cargas maiores que 80%, tal detector provocou congestionamento na rede e levou a mesma a descartar mensagens.

Carga	Período de Monitoramento (ms)					AFD
	1,0	2,0	3,0	5,0	8,0	
0%	1,35	2,31	3,31	5,31	8,31	0,61
10%	1,34	2,33	3,34	5,34	8,34	0,63
20%	1,37	2,37	3,37	5,37	8,37	0,66
30%	1,39	2,39	3,40	5,39	8,40	0,71
40%	1,42	2,41	3,41	5,40	8,41	0,78
50%	1,45	2,43	3,43	5,41	8,43	0,98
60%	1,47	2,44	3,44	5,42	8,45	1,26
70%	1,51	2,45	3,45	5,41	8,46	1,56
80%	1,59	2,47	3,46	5,41	8,47	1,94
90%	4,86	2,64	3,53	5,41	8,47	2,83

(a) Tempo de Detecção

Carga	Período de Monitoramento (ms)					AFD
	1,0	2,0	3,0	5,0	8,0	
0%	1,00	2,01	3,01	5,30	8,61	0,30
10%	0,98	1,99	2,98	4,98	7,99	0,31
20%	0,97	1,97	2,97	4,97	7,97	0,33
30%	0,95	1,96	2,96	4,95	7,96	0,37
40%	0,94	1,96	2,96	4,98	7,96	0,38
50%	0,93	1,95	2,93	4,97	7,94	0,44
60%	0,91	1,96	2,93	4,98	7,92	0,56
70%	0,93	1,97	2,91	5,01	7,91	0,69
80%	0,98	1,97	2,90	5,01	7,91	0,87
90%	1,14	2,01	2,93	4,99	7,93	1,14

(b) Duração da Falsa Suspeita

Carga	Período de Monitoramento (ms)					AFD
	1,0	2,0	3,0	5,0	8,0	
0%	13,17	34,71	63,69	1009,96	-	2,27
10%	12,46	28,06	37,76	106,73	311,30	3,40
20%	9,49	18,76	27,69	44,68	72,53	2,19
30%	8,78	15,05	27,84	37,80	70,86	1,67
40%	11,53	15,62	31,99	38,80	84,60	1,27
50%	13,79	14,79	43,05	37,36	134,29	1,05
60%	20,19	13,77	42,23	41,91	116,89	1,07
70%	18,70	13,75	60,44	49,36	171,37	1,35
80%	10,66	14,26	79,06	53,66	273,72	1,92
90%	4,08	34,15	43,83	81,67	312,00	4,04

(c) Intervalo entre Falsas Suspeitas

Carga	Período de Monitoramento (ms)					AFD
	1,0	2,0	3,0	5,0	8,0	
0%	299	120	62	2	1	1786
10%	326	144	109	64	34	1177
20%	419	215	144	93	61	1832
30%	455	264	143	100	56	2392
40%	349	258	122	108	47	3141
50%	289	270	94	108	28	3814
60%	197	289	92	89	36	3707
70%	211	292	62	86	19	2979
80%	384	281	51	71	14	2071
90%	979	110	100	51	6	979

(d) Número de Falsas Suspeitas

Figure 4. Desempenho dos detectores em termo das métricas de QoS e do número de falsas suspeitas. Em 4(c), '-' indica ausência de valor.

Do ponto de vista do intervalo entre falsas suspeitas e do número de falsas suspeitas, o detector autonômico obteve um desempenho inferior aos demais. Isto se deve por duas razões básicas. Primeiro, como o detector autonômico na média utiliza períodos de monitoramentos menores (o que pode ser derivado a partir dos tempos de detecção), o mesmo realiza mais predições, logo comente mais erros em um intervalo de tempo menor. Segundo, a visão que o detector tem do comportamento da carga no ambiente está sempre atrasada, uma vez que, é preciso esperar que uma mensagem de *heartbeat* chegue para que uma ação de controle seja executada. Sendo assim, as ações de controle de δ, realizadas pelo gestor autonômico, estão sempre atrasadas no tempo em relação a carga, o que implica que o efeito da mudança de δ acontece apenas algumas rodadas depois que variações na carga ocorrem, o que a afeta as predições do detector adaptativo.

5. Considerações Finais

O presente artigo apresentou uma proposta de detecção autonômica, baseada em teoria de controle. Tal proposta se mostrou capaz de melhorar a velocidade tanto

na detecção de defeitos quanto na recuperação do mesmo no caso de falhas de detecção. Todavia, o mesmo demonstrou-se menos confiável quando comparado com os demais detectores adotados na simulações. Vale salientar, entretanto, que o objetivo do projeto focou única e exclusivamente na melhoria do tempo de detecção e restrições relacionadas ao consumo, pelo detector, dos recursos disponíveis. Outra ponto relevante é o fato da abordagem poder ser utilizada sem considerar detalhes do ambiente no qual o detector será implementado.

Do ponto de vista da formulação de uma proposta inicial de um detector autonômico, o detector proposto apresentou um desempenho satisfatório. Entretanto, é preciso ainda observar os requisitos relacionados à confiabilidade do detector. Além disso, apesar de considerar cenários com cargas variadas, é interessante que a abordagem seja validada utilizando modelos de simulação de ambientes com redes WAN. Mais ainda, os resultados obtidos ainda carecem de validação em cenários reais. E, por fim, estratégias para validar a robustez do controle sob cenários variados precisam ser realizadas, de modo a estabelecer os limites para a estabilidade da abordagem de controle - sob variações nos parâmetros e na estrutura do modelo da planta (detector adaptativo). Esses trabalhos serão os próximos passos abordados em nossa pesquisa.

Referências

Ang, K., Chong, G., Li, Y., Ltd, Y., and Singapore, S. (2005). PID control system analysis, design, and technology. *IEEE Trans. on Control Systems Technology*, 13(4):559–576.

Bertier, M., Marin, O., and Sens, P. (2003). Performance analysis of hierarchical failure detector. In *Proc. Of The International Conf. On DSN*, pages 635–644, San-francisco,Usa. IEEE Society Press.

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 Trans. On Computer*, 51(2):561–580.

de Sá, A. S. and Macêdo, R. J. A. (2005). An adaptive failure detection approach for real-time distributed control systems over shared ethernet. In *Proc. of 18th Intern. Congress of Mechanical Engineering – Symp. Series in Mechatronics*, volume 2, pages 43–50, Ouro Preto, Brazil. COBEM2005.

Falai, L. and Bondavalli, A. (2005). Experimental evaluation of the qos failure detectors on wide area network. In *Proc. of the International Conf. On DSN*, pages 624–633. IEEE Computer Society.

Felber, P. (1998). *The Corba Object Group Service : A Service Approach to Object Groups in CORBA*. PhD thesis, Département D'Informatique, École Polytechnique Fédérale De Lausanne.

Fengyuan, R. and Chuang, L. (2003). Speed up the responsiveness of active queue management system. *IEICE Trans. on Comm. E86-B (2)*, pages 630–636.

Hellerstein, J. L., Diao, Y., Parekh, S., and Tilbury, D. M. (2004). *Feedback Control of Computing Systems*. Wiley-Interscience, Canada.

Henriksson, D. and Cervin, A. (2003). Truetime 1.13 - reference manual. Technical Report Isrn Lutfd2/Tfrt--7605--se, Dep. Of Automatic Control, Lund Institute Of Technology.

Hermant, J. and Lann, L. (2002). Fast asynchronous uniform consensus in real-time distributed systems. *IEEETC: IEEE Trans. on Computers*, 51.

Huebscher, M. C. and McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications. *ACM Comput. Surv.*, 40(3):1–28.

Jacobson, V. (1988). Congestion avoidance and control. *ACM Computer Communication Review; Proc. Of The Sigcomm '88 Symp. In Stanford, Ca, August, 1988*, 18, 4:314–329.

Macêdo, R. (2000). Failure detection in asynchronous distributed systems. In *2nd Workshop on Tests and Fault-Tolerance. pp.76-81.*

Macêdo, R. J. A. and Lima, F. (2004). Improving the quality of service of failure detectors. *Simpósio Brasileiro de Redes de Computadores.*

Majumdar, R., Ramamritham, K., Banavar, R., and Moudgalya, K. (2004). Disseminating dynamic data with qos guarantee in a wide area network: a practical control theoretic approach. *Real-Time and Embedded Technology and Applications Symp., 2004. Proc.. RTAS 2004. 10th IEEE*, pages 510–517.

Mills, K., Rose, S., Quirolgico, S., Britton, M., and Tan, C. (2004). An autonomic failure-detection algorithm. In *WOSP '04: Proceedings of the 4th international workshop on Software and performance*, pages 79–83, New York, NY, USA. ACM.

Müller, M. (2004). Performance evaluation of a failure detector using SNMP. Semester project, École Polytechnique Fédérale de Lausanne, Switzerland.

Nunes, R. C. and Jansch-Pôrto, I. (2004). Qos of timeout-based self-tuned failure detectors: the effects of the communication delay predictor and the safety margin. In *International Conf. On DSN*, pages 753–761. IEEE Computer Society.

Ogata, K. (1995). *Discrete-Time Control Systems*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 2nd edition.

Sampaio, L. M. R. and Brasileiro, F. V. (2004). Performance analysis of adaptive consensus protocols based on slowness oracles. In *Proc. of the 24th IEEE Inter. Conf. on DCS Workshops*, pages 340–346. IEEE Computer Society.

Simon, D. (2002). Analyzing control system robustness. *Potentials, IEEE*, 21(1):16–19.

The Mathworks (2002). *Matlab:The Language of Technical Computing*. Nantick, USA.

The Mathworks (2006). *Getting Started with Simulink*. Nantick, USA.

Xiong, N., Yang, Y., Chen, J., and He, Y. (2006). On the quality of service of failure detectors based on control theory. In *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, volume 1, pages 6 pp.–.