Avaliando o Impacto de Detectores de Defeitos na Estabilidade de Sistemas de Controle de Tempo Real sobre Redes Convencionais

Alírio Santos de Sá [*], **Raimundo José de Araújo Macêdo**

Laboratório de Sistemas Distribuídos – LaSiD
Programa de Pós-Graduação em Mecatrônica [†]
Universidade Federal da Bahia
Campus de Ondina, 40170-110, Salvador, Brasil

{aliriosa|macedo}@ufba.br

Resumo. *O presente artigo avalia o desempenho de um detector de defeitos adaptável, baseado em redes neurais artificiais, quando aplicado a um Sistema de Controle Distribuído de tempo real sobre uma rede Ethernet. Simulações realizadas sobre um sistema simples de controle replicado, evidenciam as vantagens da referida proposta quando contraposta com as principais abordagens adaptativas existentes. Adicionalmente, constatou-se que, sobre certas condições, a sobrecarga causada pelo uso de detectores de defeitos não prejudica o desempenho de algoritmos de controle convencionais, como PID.*

Abstract. *This paper evaluates the performance of an adaptive failure detector based on artificial neural networks intended to real-time distributed control systems over a shared Ethernet network. Simulations carried out on a simple replicated control application show the advantages of the suggested detector when compared with existing known adaptive failure detection approaches. Additionally, it has been shown that the overhead caused by the use of failure detectors in this context does not compromise the performance of conventional control algorithms such as PID.*

1. Introdução

Sistemas de Controle Discreto – aqui referenciados simplesmente por Sistemas de Controle – são sistemas que, em instantes discretos de tempo, monitoram e atuam sobre uma ou mais variáveis que compõem uma planta (i.e., qualquer objeto físico a ser controlado, como um reator químico, um automóvel, um forno, um robô etc.[25]), fazendo com que tal planta se comporte da maneira desejada [26]. Esses sistemas são exemplos típicos de sistemas de tempo real, uma vez que, devem atender a prazos ditados pela própria dinâmica do objeto físico a ser controlado. De modo geral, tais sistemas são formados por elementos sensores, que percebem o atual estado das variáveis a serem controladas; elementos controladores, responsáveis por, a partir do estado percebido, fornecer a ação de controle requerida para manter o estado das variáveis dentro do desejado; e elementos atuadores, responsáveis por capturar ação de controle e promover a efetiva atuação sobre a variável que está sendo controlada.

[*]Bolsista de Mestrado da FAPESB - Fundação de Amparo à Pesquisa do Estado da Bahia
[†]Promovido conjuntamente pelos Departamentos de Ciência da Computação e Engenharia Mecânica

O avanço das redes de computadores tem dado subsídios para que a indústria de controle e automação de processos implemente soluções de controle distribuído utilizando redes convencionais. Tais soluções, entre outras coisas, facilitam a interoperabilidade entre componentes de fabricantes distintos; diminuem a complexidade da interconexão entre componentes dos sistemas; permitem um monitoramento e supervisão remotos mais eficientes; e diminuem os custos operacionais. Entretanto, apesar de todos esses benefícios, o atraso inserido pela rede, entre a transmissão e entrega dos dados, pode influenciar no desempenho do controle realizado e tornar o projeto do sistema mais complexo.

Desafios ainda maiores são encontrados quando se deseja implementar aplicações críticas de controle distribuído, as quais necessitam de mecanismos de tolerância à falhas para garantir seu funcionamento contínuo. Para a implementação dessas aplicações sobre redes convencionais é essencial promover soluções de detecção de defeitos adaptáveis, seja para ativar mecanismos de recuperação, seja para permitir a identificação da réplica defeituosa ou para possibilitar uma reconfiguração do sistema. Os algoritmos de adaptação utilizados na detecção devem contornar ou minimizar os possíveis efeitos do atraso não determinístico imposto pela rede de comunicação. Mecanismos de detecção adaptável com *QoS* têm sido discutidos em alguns trabalhos, como em [7, 2, 3, 21]. Essas propostas tratam a detecção de defeitos para sistemas distribuídos convencionais, não considerando aplicações de controle distribuído. A aplicabilidade dos algoritmos convencionais de tolerância a falhas foi analisada por [13], onde os autores demonstram que detecção de defeitos para sistemas convencionais parcialmente síncronos pode ser utilizada para projetar sistemas distribuídos de tempo real críticos, através de um método denominado *Ligação Tardia*(*Late Binding*). Todavia, em [13], não são utilizadas redes de comunicação convencionais como a *Ethernet/CSMA-CD*, por exemplo.

Devido ao seu baixo custo e suas altas taxas nominais de transmissão, a *Ethernet* [30] se mostra bastante atraente para uso em sistemas de controle de tempo real. No entanto, o não determinismo inerente ao protocolo *CSMA/CD* pode levar ao não cumprimento dos limites temporais das aplicações críticas. O não cumprimento dos prazos pode ocasionar perda na qualidade do controle realizado, ou ainda resultar em um controle ineficiente (instável) [19], o que , por sua vez, pode trazer conseqüências desastrosas para aplicações críticas de controle. Em [6] define-se o conceito de *margem de jitter* para avaliar a implementação de sistemas de controle de tempo real sujeitos a atrasos de entrada e saída provocado pela politica de escalonamento de tarefas. Em [28] o conceito é estendido e usado para avaliar a implementação de sistemas de controle sobre redes *CAN*. Esses trabalhos focam na implementação do sistemas de controle e não discutem como mecanismos de tolerância a falhas podem ser implementados.

Nesse contexto, o presente artigo traz as seguintes contribuições: propõe uma abordagem de detecção adaptável, utilizando Redes Neurais Artificiais (*RNA*), capaz de se adequar a variações do atraso na comunicação sobre redes *Shared-Bus-Ethernet*; contrapõe a abordagem proposta aos detectores adaptáveis com *QoS* para sistemas distribuídos convencionais; demonstra, através de simulações, a viabilidade do uso de tal abordagem em um ambiente com Sistema de Controle Distribuído de tempo real simples e em um ambiente com o barramento *Ethernet* sendo compartilhado com múltiplos Sistemas de Controle; avalia, usando o conceito de *margem de jitter*, o impacto da

implementação de detectores de defeitos no desempenho do sistema de controle[1].

O presente artigo está organizado da seguinte forma. A seção 2 apresenta as principais abordagens de detecção adaptáveis com *QoS* para sistemas distribuídos convencionais. A seção 3 discute o impacto da implementação de detectores de defeitos para sistemas de controle distribuídos de missão-crítica sujeitos a atrasos e *jitter*. As seções 4, 5 e 6 discutem, respectivamente, a abordagem de detecção proposta, as simulações realizadas e as conclusões.

2. Abordagens Adaptáveis de Detecção de Defeitos

As abordagens de detecção apresentadas utilizam, na avaliação do serviço de detecção de defeitos, as métricas propostas por [7]. Tais métricas avaliam a rapidez na detecção de falhas e a capacidade do detector de evitar falsas suspeitas. As principais métricas são: *Tempo de Detecção* (T_D), intervalo de tempo entre o instante no qual o dispositivo falhou e o instante no qual este dispositivo falho é suspeito pelo detector; *Duração da Falsa Suspeita* (T_M), tempo necessário para se corrigir uma falsa suspeita; *Tempo entre Falsas Suspeitas* (T_{MR}), tempo esperado entre duas falsas suspeitas consecutivas.

Todas as abordagens discutidas usam o monitoramento baseado em *heartbeats* comentado por [7], no qual, são considerados dois dispositivos p e q, onde q possui um módulo detector embutido e monitora falhas de p. A cada Δ^i unidades de tempo, p envia para q uma mensagem, sequencialmente assinalada e denotada por *heartbeat*, informando que está funcionando corretamente. Para cada *heartbeat* (m_k^{hb}) recebido, q calcula o intervalo de tempo (Δ^{to}) necessário para a chegada do próximo *heartbeat* (m_{k+1}^{hb}). Se m_{k+1}^{hb} não chega dentro de Δ^{to} unidades de tempo, q coloca p em sua lista de suspeitos. Por outro lado, caso q receba um *heartbeat* com o seqüencial igual ou superior ao seqüencial do *heartbeat* esperado, q remove p de sua lista de suspeitos. p envia mensagens m^{hb} em instantes denotados por σ, desta forma: m_k^{hb} é enviada no instante σ_k, m_{k+1}^{hb} em σ_{k+1} e assim sucessivamente. Para quaisquer dois instantes consecutivos σ_k e σ_{k+1}, tem-se $\sigma_{k+1} - \sigma_k = \Delta^i$. Os instantes de chegada das mensagens m^{hb} são denotados por A, ou seja: m_k^{hb} chega em A_k, m_{k+1}^{hb} em A_{k+1} e assim por diante. Sendo d_k o tempo de viagem de m_k^{hb}, conclui-se que: $A_k = \sigma_k + d_k$. Portanto, cada *heartbeat* está *envelhecido* de pelo menos d unidades de tempo.

Quando não há variação no atraso nem perda de mensagens na rede, o menor intervalo de tempo no qual q pode começar a suspeitar da falha de p com segurança não pode ser menor que $T_D^{min} = d + \Delta^i$, T_D^{min} é o tempo mínimo de detecção. Se variações no atraso são consideradas, tem-se $A_{k+1} - A_k = \Delta^i + j_k$ e $j_k = d_{k+1} - d_k$ (j_k representa a variação observada entre o atraso na entrega da mensagem m_k e m_{k+1}).

Quando os tempos de viagem das mensagens não são conhecidos e não há relógios sincronizados, as estimativas do detector não são precisas e nem é possível estimar T_D com precisão. Para a estimativa EA_k realizada pelo detector para o instante de chegada de m_k^{hb}, quanto mais próximo EA_k estiver de A_k, menor será T_D. A condição $EA_k \geq A_k$ deve ser satisfeita para que o detector evite falsas suspeitas. Assim, a relação entre EA e A é uma indicação do desempenho do detector em termos de tempo de detecção. Uma vez

[1]resultados preliminares deste trabalho foram discutidos em [9] e em [8]. Todavia, em [9] é analisada apenas a implementação dos detectores para um único sistema de controle e em [8] não são investigadas a estabilidade e a qualidade do desempenho do sistema de controle em questão.

que variações extras no atraso podem provocar sub-estimativas, utiliza-se uma margem de segurança (α) que compense variações não previstas no atraso da rede [14, 7]. Desta forma, o marco estimado no tempo FP (*Freshness Point* [7]) para chegada do próximo *heartbeat* é definido por $FP_{k+1} = EA_{k+1} + \alpha_{k+1}$. Sendo assim, A adaptabilidade do detector consiste em ajustar EA e α [24]. Logo, recebido m_k^{hb} em A_k, quanto mais precisa a estimativa de EA_{k+1} e α_{k+1} mais próximo FP_{k+1} estará de A_{k+1}.

O autor de [14] sugere uma estimativa para predição de atraso de modo a evitar que retransmissões desnecessárias de mensagens colaborem para o congestionamento na rede IP. Tal algoritmo é descrito a seguir. Sendo d_k^m e d_k^c, respectivamente, os atrasos medido e estimado no instante k, o algoritmo realiza uma estimava de d_{k+1}^c baseado em d_k^c e na confiança μ atribuída ao desvio (e_k) entre d_k^c e d_k^m. Dessa forma, $e_k = d_k^m - d_k^c$ e $d_{k+1}^c = d_k^c + \mu.e_k$. Considerando a existência de variações adicionais no atraso da rede, a nova estimativa para o atraso (Δ_{k+1}^{to}) na chegada de uma mensagem é $\Delta_{k+1}^{to} = \beta.d_{k+1}^c - \phi.v_{k+1}$, onde ϕ e β são, respectivamente, a confiança na variação do atraso e na estimativa do atraso calculado. v_k representa a variação no atraso da rede medida no instante k e pode ser calculado por $v_{k+1} = v_k - \mu.(|e_k| - v_k)$. Por fim, presume-se que a próxima mensagem chegará em $FP_{k+1} = A_k + \Delta_{k+1}^{to}$. Os autores de [2] propuseram um detector adaptativo implementado em duas camadas, onde a camada inferior provê o serviço básico de detecção baseado nas métricas de *QoS* propostos por [7], enquanto que a camada superior ajusta a *QoS* da camada inferior de acordo com as necessidades da aplicação. A estimativa de EA realizada na camada inferior é baseada na proposta de [7], onde $EA_{k+1} = \frac{1}{n}[A_k - A_{k-n-1} + (k+1).\Delta^i]$. Para as n mensagens iniciais utiliza-se: $U_{k+1} = \frac{A_k}{k+1}.\frac{k.U_k}{k+1}$, onde $U_1 = A_0$; e calcula-se $EA_{k+1} = U_{k+1} + \frac{k+1}{2}.\Delta^i$. O tempo esperado para m_{k+1}^{hb} é $FP_{k+1} = EA_{k+1} + \alpha_{k+1}$. O ajuste de α utiliza o algoritmo proposto em [14], assim os autores calculam o erro e a margem de segurança por $e_k = A_k - EA_k - d_k$ e $\alpha_{k+1} = \beta.d_{k+1} - \phi.v_{k+1}$.

3. Estabilidade e Desempenho do Controle Distribuído

No projeto de sistemas de controle de tempo real precisa-se lidar com incertezas inerentes à plataforma de computacão[6, 5] e de comunicação[18, 19]. Tais incertezas estão relacionadas às variações no tempo de execução das tarefas (atraso de computação) e no envio e entrega das mensagens pelo subsistema de comunicação (atraso de comunicação). Essas variações podem alterar o comportamento do controle, degradando o desempenho e podendo levar o sistema a comportar-se de forma instável [23, 22]. O projeto clássico do sistema de controle, muitas vezes não leva em consideração a existência de tais incertezas, ou ainda pressupõe que essas são muito pequenas para serem consideradas. Na prática essas incertezas podem afetar consideralvemente o sistema, principalmente quando o mesmo é implementado como *controle em rede* (*NCS, Networked Control System*) ou como um *controle distribuído* (*DCS, Distributed Control System*). Essas variações de tempo implicam diretamente em variações nos instantes de atuação e, mesmo quando não levam à instabilidade, podem alterar o desempenho do controle realizado.

O problema torna-se ainda mais complexo quando aplicações de controle de *missão-crítica* precisam ser implementadas. Essas aplicações precisam de mecanismos que atribuam um maior grau de confiaça ao funcionamento, habilitando o sistema com a capacidade de evitar ou tolerar a existência de falhas. Na ocorrência de falhas em componentes do sistema, os mecanismos de tolerância a falhas devem detectar o erro, localizar

e confinar a falha e promover a recuperação e ou reconfiguração do sistema [15]. Assim sendo, a adição desses mecanismos pode incrementar consideravelmente o atraso computacional e de comunicação, podendo fazer com que o sistema viole as condições necessárias para a garantia da estabilidade [29, 17, 10].

Medidas clássicas como a *margem de fase* e a *margem de ganho* são usadas para medir a estabilidade e descrever a sensibilidade da malha de controle quando a planta apresenta distúrbios em seu comportamento [25, 6]. Entretanto, tais medidas são definidas sem levar em consideração a existência de incertezas provocadas pela plataforma de computação e comunicação, ou falhas de dispositivos. Diversos trabalhos têm sido realizados para promover algoritmos mais eficientes [16, 20, 4, 23], bem como alguns outros trabalhos visam o desenvolvimento de metodologias de *codesign* para o desenvolvimento de controle em tempo real [23, 22, 5, 28] diminuindo a lacuna existente entre o projeto do controle e de sua implementação como sistema de tempo real.

A *margem de atraso* e a *margem de jitter* são conceitos usados para auxiliar a implementação e a qualificação de sistemas de controle de tempo real sob atrasos de entrada e saída variados[6]. As definições de tais conceitos são apresentadas a seguir.

Definição 3.1 (Margem de Atraso) *Para um sistema de controle de tempo real, a margem de atraso é definida pelo maior número L_m para o qual a estabilidade é garantida assumindo um atraso constante $\Delta = L_m$ [6]. Em sistemas de controle contínuos, a margem de atraso pode ser calculador por $L_m = \phi_m/\omega_{cp}$, onde ϕ_m e ω_{cp} representam, respectivamente, a margem de fase e a freqüência de cruzamento da fase [25].*

Definição 3.2 (Margem de Jitter) *Para um sistema de controle de tempo real, a margem de jitter é definida como o maior número $J_m(L)$ para o qual a estabilidade é garantida para qualquer atraso variante no tempo $\Delta \in [L, L + J_m(L)]$ [6]. A análise usando tal conceito deve considerar uma planta atrasada no tempo $P(s)_{delayed} = P(s)e^{-sL}$ e $N = J/h$ [16].*

Assim, segundo [6], a estabilidade do sistema de controle de tempo real, em um ambiente com atraso constante L e *jitter* J, será garantida se $J_m(L) > J$.

Em um ambiente com atraso e *jitter* o conceito de *margem de atraso* apresenta-se de acordo com a definição a seguir.

Definição 3.3 (Margem de Atraso para sistemas com atraso e *jitter*) *Assumindo um atraso constante L e jitter J, a margem de atraso é definido como o maior número L_m para o qual a estabilidade da malha fechada de controle com jitter é garantida para qualquer atraso variante no tempo $\Delta \in [L + L_m, L + L_m + J]$. Logo, a margem de atraso será dada pelo menor L_m que satisfaça $J_m(L + L_m) = J$.*

A partir da definição anterior, pode-se definir o conceito de *margem de atraso aparente*, a qual será usada como medida de desempenho para sistemas de controle de tempo real sob atraso e *jitter*.

Definição 3.4 (Margem de Fase Aparente) *Assumindo um atraso constante L e o jitter J, a margem de fase aparente é definida como o maior número $\widehat{\phi}_m$ para o qual a estabilidade de um sistema de malha fechada com jitter pode ser garantida para qualquer atraso variante no tempo $\Delta \in \left[L + \frac{\widehat{\phi}_m}{\omega_{cp}}, L + \frac{\widehat{\phi}_m}{\omega_{cp}} + J\right]$, onde ω_{cp} representa a frequência de cruzamento da fase. A estabilidade do sistema será garantida se $\widehat{\phi}_m > 0^o$ [6].*

A relação entre a *margem de fase aparente* e a *margem de fase* pode ser usada como uma medida da qualidade do desempenho do controle (*QoP, Quality of Performance*), assim $QoP = \frac{\widehat{\phi}_m}{\phi_m}$ [6].

O arcabouço teórico, do qual o conceito de *margem de jitter* foi derivido, traz apenas condições *suficientes* para a garantia da estabilidade, portanto, não necessariamente o sistema será instável se o atraso for maior que a *margem de jitter*. O contrário, no entanto, sempre é verdadeiro, ou seja, se a *margem de jitter* é atendida o sistema é estável.

3.1. Analisando a Influência do Detector de Defeitos na Estabilidade do Sistema

Para analisar a implementação do detector adota-se o modelo a seguir. Considera-se a existência de um sensor (nd^{sns}), um atuador (nd^{atd}) e um controlador (nd^{ctrl}) sobre os quais executam sistemas operacionais de tempo real. Estes elementos utilizam um subsistema de comunicação para interagir via troca de mensagens e colaboram para compor um sistema de controle distribuído. Em nd^{sns} roda uma tarefa periódica de aquisição de dados (τ^{sns}). Cada amostra coletada por τ^{sns} é enviada para o elemento nd^{ctrl}, onde a tarefa de controle (τ^{ctrl}) é acionada no instante em que a amostra é recebida. De posse da amostra, τ^{ctrl} executa um algoritmo de controle *PID* [25] e então envia a informação de controle para nd^{atd}, o qual ativa τ^{atd} para realizar a atuação sobre o objeto controlado, um motor servo DC definido pela equação de transferência $G(s) = 1000/(s^2+s)$. Assume-se a possibilidade de falha por *crash* de nd^{ctrl}, e por este motivo esse elemento é replicado, formando um esquema redundante para tolerar uma única falha. As réplicas (nd_p^{ctrl} e nd_p^{ctrl}) têm por responsabilidade: receber as mensagens oriundas de nd^{sns}; executar o algoritmo de controle; e enviar a ação de controle à nd^{atd}. nd_s^{ctrl} somente enviará sua ação de controle quando uma falha em nd_p^{ctrl} for detectada. No controlador secundário existe um detector de defeitos embutido para verificar falhas em nd_p^{ctrl} (figura 1). Na presença de falhas de nd_p^{ctrl} existe um algoritmo que permite que nd_s^{ctrl} assuma o controle, garantindo a continuidade do serviço.

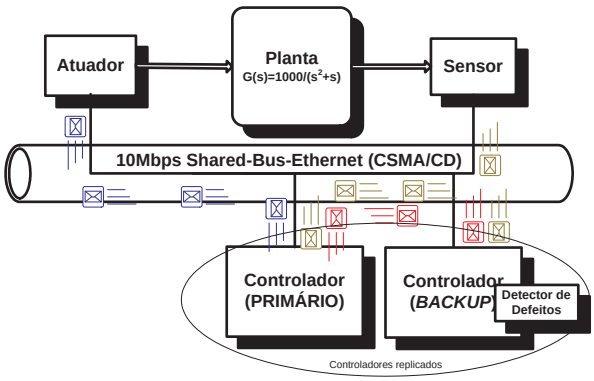

Figure 1. Modelo de sistema de controle distribuído de missão-crítica

Para tal modelo o atraso na atuação (d^{atd}) pode ser visto como o somatório de duas componentes $d^{atd} = d^{sns} + d^{ctrl}$. Onde d^{sns} e d^{ctrl} representam, respectivamente, o atraso de aquisição e o atraso da execução da ação de controle. d^{sns} representa o intervalo de tempo necessário para que o sensor perceba o atual estado da planta e entregue (através da rede *Ethernet*) o valor coletado ao elemento controlador primário. d^{ctrl} corresponde

ao intervalo de tempo necessário para que o algoritmo de controle seja executado e a acão de controle correspondente seja entregue e executada pelo elemento atuador. Os elementos sensor e controladores concorrem pelo uso do canal de comunicação, assim, existe a possibilidade da ocorrência de colisões e, portanto, os atrasos d^{atd} e d^{ctrl} podem assumir valores não determinísticos.

Para que a estabilidade do sistema de controle possa ser garantida, é *suficiente* que o atraso na atuação seja menor que o maior atraso permitido para o sistema, $d^{atd} < L+Jm(L)$. Onde o atraso L será ditado pelo somatório do tempo mínimo de transferência de mensagens na rede e pelo atraso de computação nos elementos sensor, controlador e atuador.

Caso ocorra uma falha em nd_p^{ctrl}, o chaveamento entre os controladores primário e secundário deve ocorrer em um tempo tal que garanta a estabilidade do controle. Por conta disso, o atraso na entrega provocado por tal chaveamento t_{TF} deve ser limitado pelo atraso máximo permitido pelo sistema, ou seja, $t_{TF} < L + Jm(L)$. O intervalo t_{TF} deve acomodar o tempo necessário para a detecção da falha mais o tempo necessário para a ativação e atuação do controlador secundário nd_s^{ctrl}. Assim, $t_{TF} = t_D + t_R$, onde t_D representa o tempo de detecção e t_R o tempo de reconfiguração do sistema.

t_D está diretamente ligado ao atraso do canal de comunicação e à taxa de *heartbeats* Δ^i (conforme discutido na seção anterior). Apesar do atraso de comunicação ser influenciado por Δ^i (quanto menor Δ^i maior o atraso de comunicação), apenas a taxa de *heartbeat* está ligada diretamente a implementação do detector de defeitos. Dessa forma, durante o projeto do algoritmo de detecção de defeitos deve-se selecionar uma taxa de *hearbeat* tal que minimize t_D e garanta um atraso de comunicação que permita a estabilidade do sistema de controle.

4. Detecção Baseada em RNA para Sistema de Controle Distribuídos

A predição baseada em RNA faz uso de uma Rede Neural do tipo MLP (*Multilayer Perceptron* [11]) com quatro camadas: três neurônios na camada de entrada; um neurônio na camada de saída; e duas camadas ocultas com trinta e dez neurônios respectivamente. No treinamento da *RNA* utilizou-se o algoritmo *resilient propagation* [27] para a atualização dos pesos sinápticos de acordo com uma taxa de aprendizado Δ adaptativa. Nessa fase, utilizou-se as seguintes configurações: $\Delta = 0,1$; fatores de atualizações de Δ para as derivadas parciais negativas e positivas dos pesos sinápticos, $\eta^- = 0,5$ e $\eta^+ = 1,2$; treinamento com $5x10^4$ épocas; erro mínimo igual a $0,0$; e gradiente mínimo igual $1x10^{-12}$, como em [27]. As variáveis de entrada para a *RNA* são: *Atraso entre chegadas*, representa o último atraso verificado entre dois *heartbeat* consecutivos ($A_k - A_{k-1}$); *Variação do atraso* (j_k); e *Taxa de heartbeat* (Δ^i). De posse dessas variáveis a *RNA* fornece o valor estimado para o intervalo de tempo (Ω) entre a chegada de dois *heartbeats* consecutivos. Assim, estima-se a chegada de m_{k+1}^{hb} em $EA_{k+1} = EA_k + \Omega_{k+1}$.

5. Simulações Realizadas

As simulações analisam o desempenho dos detectores em termos do *valor médio* (T_D^{mean}, T_M^{mean} e T_{MR}^{mean}) e do *desvio padrão* (T_D^{std}, T_M^{std} e T_{MR}^{std}) das métricas de *QoS*. Além das métricas de *QoS*, verifica-se o número de falsas suspeitas (N^{fs}) cometidas por cada detector.

Na primeira simulação, as análises observam a *QoS* em ambientes com atraso moderado, onde a rede é utilizada apenas para transferir mensagens entre os dispositivos do sistema de controle. Esse ambiente tenta refletir o comportamento do detector em um sistema de controle distribuído tradicional. Na segunda simulação, as análises observam o desempenho dos algoritmos para um ambiente com barramento compartilhando entre múltiplos sistemas de controle. As simulações comparam o desempenho dos algoritmos de [14] e [2] com a abordagem proposta. As experimentações utilizam $\Delta^i = \{1ms, 5ms, 10ms, 50ms, 100ms, 200ms\}$, $\alpha_0 = 0ms$ e um número de mensagens $N = 520$ por Δ^i (sendo 52 dessas mensagens usadas na inicialização dos algoritmos). Utiliza-se $\beta = 1$, $\phi = 2$ e $\mu = 0,1$, como propostos por [2] e [14].

Dada à dificuldade em se medir T_D com precisão (ver seção 2), o mesmo é calculado por $T_D(k+1) = EA_{k+1} - A_k$, onde $T_D(k+1)$ representa o tempo de detecção observado para uma falha antes do envio de m^{hb}_{k+1}. Para possibilitar uma análise mais acurada dos tempos de detecção sugeridos pelos preditores avaliados, são apresentados os intervalos entre *heartbeats* consecutivos $A_{k+1} - A_k$. Para execução das simulações utilizou-se o *TrueTime* versão 1.3 [12], um *Toolbox* para simulação de sistemas de tempo real distribuídos, disponível na ferramenta *Simulink/Matlab* [31]. No *TrueTime* cada dispositivo (nd^{sns}, nd^{atd} etc.) representa uma estação com sistema operacional com *kernel* multitarefa e de tempo real. Selecionou-se, no simulador, a rede de barramento compartilhado *Ethernet/CSMA-CD* (*shared-bus-Ethernet*) de $10Mbps$ e os dispositivos foram configurados para trabalhar com uma política de escalonamento de tarefas baseada em prioridades fixas. A periodicidade, o período de ativação T e o prazo de computação D das tarefas no *TrueTime* são: $\tau^{ctrl} : \{esporadica, D = 6ms\}$; $\tau^{sns} : \{periodica, T = 10ms, D = 10ms\}$; e $\tau^{atd} : \{esporadica, D = 20ms\}$.

5.1. Avaliando o Desempenho das Abordagens de Detecção de Defeitos

As tabelas 1 e 2 apresentam as avaliações realizadas para verificar o desempenho dos algoritmos de detecção de defeitos. Nessas tabelas, $A_{k+1} - A_k$ representa o tempo entre *heartbeats* consecutivos que chegam em nd^{ctrl}_s e serve como referência para analisar o quão próximo as estimativas de EA_k realizadas pelos detectores estão do valor real observado por nd^{ctrl}_s. Nas tabelas, os valores de tempo são expressos em milisegundos.

Table 1. Avaliação dos detectores em um ambiente com um único sistema

Δ^i	$A_{k+1} - A_k$		Detector	T_D^{mean}	T_D^{std}	T_{MR}^{mean}	T_{MR}^{std}	T_M^{mean}	T_M^{std}	N^{fs}
	$mean$	std								
1,00	1,00	0,04	Jacobson	1,04	0,01	10,00	0,00	0,07	0,00	1365
			Bertier	6,00	0,02	-	-	-	-	0
			RNA	1,09	0,03	-	-	-	-	0
5,00	5,00	0,00	Jacobson	5,00	0,00	-	-	-	-	0
			Bertier	30,00	0,00	-	-	-	-	0
			RNA	5,45	0,00	-	-	-	-	0
100,00	100,00	0,00	Jacobson	100,11	0,50	857,14	1075,44	0,00	0,00	22
			Bertier	160,51	215,67	1040,00	2101,90	-	-	6
			RNA	100,00	0,00	-	-	-	-	0
200,00	200,00	0,00	Jacobson	200,23	1,00	1575,00	2304,86	0,00	0,00	25
			Bertier	321,42	432,82	-	-	-	-	1
			RNA	200,00	0,00	-	-	-	-	0

Nos resultados obtidos na tabela 1, observa-se que o algoritmo baseado em *RNA* possui um desempenho melhor que os demais em todos os parâmetros avaliados. O algo-

ritmo de *Jacobson* possui uma precisão ruim, uma vez que comete muitas falsas suspeitas. Contudo, este e o algoritmo de *Bertier* corrigem suas falsas suspeitas muito rapidamente ($T_M^{mean} \approx 0,00$). O algoritmo de *Bertier* apresentou os piores tempos de detecção, todavia, obteve um melhor desempenho, quando comparado ao algoritmo de *Jacobson*, em termos de T_{MR}^{mean} e N^{fs}. Os valores elevados de T_D^{mean} apresentados pelo algoritmo de *Bertier* nas simulações se dão por conta do procedimento de inicialização de tal detector. Após a fase de inicialização o detector passa a produzir estimativas muito elevadas, mas durante a fase de execução esses valores diminuem gradativamente a cada estimativa.

Table 2. Avaliação dos detectores em um ambiente com multiplos sistemas

Sistemas	Atraso Entre Chegada $A_{k+1} - A_k$		QoS	Detectores					
				Bertier		Jacobson		RNA	
	$mean$	std		$mean$	std	$mean$	std	$mean$	std
Sistema 1	50,00	0,71	T_D	68,21	56,49	51,09	0,67	56,63	8,13
			T_M	0,58	0,50	0,66	0,61	4,56	6,67
			T_{MR}	458,13	404,16	561,68	446,24	1706,25	1154,49
			N^{fs}	32		31		9	
Sistema 2	50,00	0,93	T_D	68,27	56,42	51,24	0,81	56,35	13,02
			T_M	0,85	1,12	0,74	1,07	14,40	28,69
			T_{MR}	571,33	442,29	538,24	385,71	1253,43	1388,69
			N^{fs}	29		35		16	
Sistema 3	50,00	0,77	T_D	68,22	56,47	51,11	0,69	56,08	8,60
			T_M	0,77	0,68	0,65	0,52	6,75	8,53
			T_{MR}	441,44	399,50	505,41	376,98	1082,34	880,36
			N^{fs}	36		38		18	
Sistema 4	50,00	0,80	T_D	68,31	56,44	51,19	0,72	56,95	12,92
			T_M	0,76	0,71	0,71	0,68	5,17	4,70
			T_{MR}	485,02	449,27	519,74	431,41	1175,06	1226,94
			N^{fs}	31		34		17	
Sistema 5	50,00	0,70	T_D	68,17	56,48	51,02	0,67	56,06	7,77
			T_M	0,68	0,75	0,69	0,63	8,84	6,48
			T_{MR}	510,33	345,43	666,60	469,45	1312,53	1617,81
			N^{fs}	30		28		9	
Sistema 6	50,00	0,72	T_D	68,24	56,44	51,09	0,64	56,53	9,96
			T_M	0,71	0,55	0,66	0,55	8,23	8,29
			T_{MR}	426,50	310,09	546,91	329,02	1036,37	993,83
			N^{fs}	35		33		12	

Os dados apresentados da tabela 2, foram obtidos em um ambiente onde o barramento é compartilhado com múltiplos sistemas de controle. Nessa simulação, cada sistema é composto por 4 elementos (um sensor, um atuador e dois controladores). Os detectores utilizam um período de emissão de *heartbeats* de $50ms$. Observa-se que, em geral, o algoritmo de $Jacobson$ obteve o melhor desempenho em termos de T_D^{mean} e T_M^{mean}, significando que o detector com esse algoritmo realiza uma detecção mais rápida e corrige mais rapidamente uma falsa suspeita. Entretanto, a RNA mostrou-se mais confiável apresentando um número muito menor de falsas suspeitas N^{fs} e um maior T_{MR}^{mean} em relação aos outros dois algoritmos. Comparando a RNA com o algoritmo de *Bertier*, pode-se notar que essa, além de cometer menos erros, possui um menor tempo de detecção. O algoritmo de *Bertier*, entretanto, consegue corrigir mais rapidamente suas falsas suspeitas.

5.2. Avaliando o Impacto no Desempenho do Controle

Para avaliação do impacto da implementação do detector de defeitos no sistema de controle utilizou-se J_m e $\widehat{\phi}_m$. Além disso, observou-se dois índices clássicos de desempenho: a Integral do Erro Absoluto (*IAE, Integral Absolute Error*) e a Integral do Erro Quadrático

(*ISE, Integral Square Error*)[2]. Esses índices de desempenho foram obtidos seguindo as equações a seguir: $IAE = \sum_{i=0}^{n} |r_i - y_i|$ e $IAE = \sum_{i=0}^{n} (r_i - y_i)^2$, onde r_i e y_i representam o valor desejado e a estado da planta, respectivamente.

Na tabela 3, calculou-se a *margem de jitter* considerando um atraso na atuação constante de $112,2\mu s$. Esse atraso é equivalente ao somatório do tempo mínimo de execução do processo de sensoriamento (configurado na simulação para $5\mu s$), mais o tempo mínimo para execução da tarefa de controle (configurado na simulação para $5\mu s$), mais os tempos mínimos de transferência de uma mensagem entre sensor-controlador e controlador-atuador ($2 * 51,2\mu s = 102,2\mu s$). Assim, a *margem de jitter* calculada foi de $J_m(112,2\mu s) = 3,7ms$. Avaliando a tabela 3 pode-se observar que o *jitter* J inserido no sistema pelo menor período de emissão de *heartbeat* utilizado ($\Delta^i = 1ms$) é de $1,1ms$ o que implica que o sistema é estável: $J_m(112,2\mu s) = 3,7ms > 1,1ms$. Apesar de $\widehat{\phi}_m$ e J se manterem aproximadamente iguais para todos os períodos de emissão de *heartbeats* utilizados, os índices de desempenho IAE e ISE apresentaram uma queda com o incremento de Δ^i, demonstrando uma melhora no desempenho do sistema de controle. Como pode-se observar, para manter o tempo de atuação dentro dos limites de estabilidade (margem de jitter menor que $3,7ms$), o período de emissão de *heartbeat* tem que ser suficientemente pequeno para acomodar o periodo de recuperação - ou seja, o tempo necessário para que o controlador secundário passe a controlar a planta. Na tabela 3, verifica-se que um período de $1ms$ para a emissão de heartbeat possibilitaria um tempo de recuperação de aproximadamente $2,7ms$, sem que a estabilidade seja comprometida. Deve-se observar, no entanto, que essa análise é particular da planta em questão, um motor *servo DC* controlado por um controlador PID.

Table 3. Avaliação do desempenho do sistema de controle sob diferentes Δ^i.

L_m	7,8	J_m	3,7	ϕ_m	34^o
Δ^i	J	$\widehat{\phi}_m$	$\frac{\widehat{\phi}_m}{\phi_m}$	IAE	ISE
$1,00$	1,1	$14,34^o$	0,42	$2,79 * 10^3$	$1,79 * 10^3$
$5,00$	1,1	$14,34^o$	0,42	$0,97 * 10^3$	$0,64 * 10^3$
$10,00$	1,1	$14,34^o$	0,42	$0,69 * 10^3$	$0,46 * 10^3$
$100,00$	1,1	$14,34^o$	0,42	$0,52 * 10^3$	$0,37 * 10^3$

6. Conclusões e Trabalhos Futuros

Os resultados obtidos demonstram que, em um ambiente de tráfego moderado, a implementação do detector de defeitos não influencia o desempenho de um sistema de controle típico (representado neste artigo por motor servo DC sendo controlado por um controlador PID), mesmo quando se considera um barramento compartilhado por multiplos sistemas, o que reforça os resultados obtidos em [8] e [9]. Além disso, os resultados ressaltam a relação de compromisso existente entre confiabilidade e qualidade de controle durante o projeto de sistemas de controle de missão-crítica. Garantir a confiabilidade e maximizar o desempenho do controle são dois requisitos conflitantes, haja vista que para se acionar mais rapidamente os mecanismos de tolerância a falhas é necessário que o detector de defeitos possua baixos tempos de detecção. No entanto, isso exige a utilização

[2]Ver [25] para maiores detalhes sobre esses índices de desempeho.

de menores períodos de emissão de heartbeats, o que provoca um aumento no atraso de comunicação e, consequentemente, degrada o desempenho do controle.

Adicionalmente, este artigo apresentou uma proposta de detecção de defeitos baseada em rede neural para um sistema de controle distribuído. Demonstrando, através das simulações, as vantagens da utilização desta proposta em relação às abordagens de detecção para sistemas distribuídos convencionais existentes na literatura.

Na implementação dos algoritmos do sistema (*RNA*, analisador de tráfego, controle *PID* e detectores) foi utilizada a linguagem de *scripts* do *MatLab* em conjunto com as funções para implementação de tarefas de tempo real disponibilizadas pelo *TrueTime*.

Atualmente, encontra-se em desenvolvimento a implementação do algoritmo de detecção proposto em uma plataforma distribuída real para Controle e Supervisão de plantas industriais, denominada *ARCOS*[1]. As análises estão sendo estendidas para contempla diferentes redes de controle, como *Switched-Ethernet*, *DeviceNet* e *ControlNet*.

Referências

[1] S. S. Andrade and R. J. A. Macêdo. A component-based real-time architecture for distributed supervision and control applications. In *10th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, volume I, pages 19–22, Catania, Italy, September 2005.

[2] M. Bertier, O. Marin, and P. Sens. Implementation and performance evaluation of an adaptable failure detector. In *Proc. Of The Int. Conf. On DSN*, pages 354–363, Washington, Dc, Jun 2002.

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

[4] G. Buttazzo, M. Velasco, P. Martí, and G. Fohler. Managing quality-of-control performance under overload conditions. In *Proc. of the 16th Euromicro Conf. on Real-Time Systems*, pages 53–60, Washington, DC, USA, 2004. IEEE Computer Society.

[5] A. Cervin. Merging real-time and control theory for improving the performance of embedded control systems. Research report, Department of Computer Engineering and Systems Science, University of Pavia, Pavia, Italy, Set 2004.

[6] A. Cervin, B. Lincoln, J. Eker, K.-E. Arzen, and G. Buttazzo. The jitter margin and its application in the design of real-time control systems. In *Proc. of the 10th Int. Conf. on Real-Time and Embedded Computing Systems and Applications*, pages 1–9, Goteborg, Sweden, aug 2004.

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

[8] A. S. de Sá and R. J. A. Macêdo. An adaptive failure detection approach for real-time distributed control systems over shared ethernet. In *Proc. of 18th Int. Congress of Mechanical Engineering*, Ouro Preto, Brazil, nov 2005. COBEM2005.

[9] A. S. de Sá and R. J. A. Macêdo. Detectores de defeitos adaptáveis para sistemas de controle distribuídos de tempo-real sobre redes ethernet. In *Proc. of VII Brazilian Workshop of Real-Time Systems*, pages 65–68, Fortaleza, Brazil, May 2005.

[10] C. R. Elks, J. Bechta Dugan, and B. W. Johnson. Reliability analysis of hard real-time systems in the presence of controller malfunctions. In *Proc. of the XVIII Reliability and Maintainability Symposium*, pages 58–64, Los Angeles, CA, jan 2000. IEEE Computer Society Press.

[11] S. Haykin. *Neural Networks; A Comprehensive Foundation*. MACMillan, New York, 1st edition, 1994.

[12] D. Henriksson and A. Cervin. Truetime 1.13 - reference manual. Technical Report Isrn Lutfd2/Tfrt--7605--se, Department Of Automatic Control, Lund Institute Of Technology, Oct 2003.

[13] J.F. Hermant and Le Lann. Fast asynchronous uniform consensus in real-time distributed systems. *IEEETC: IEEE Trans. on Computers*, 51, 2002.

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

[15] P. Jalote. *Fault Tolerance In Distributed Systems*. Prentice Hall, New Jersey, 1994.

[16] C.-Y. Kao and B. Lincoln. Simple stability criteria for systems with time-varying delays, March18 2004.

[17] H. Kim and K. G. Shin. On the maximum feedback delay in a linear/nonlinear control system with input disturbances caused by controller-computer failures. *IEEE Trans. on Control Systems Technology*, 2(2):110–122, 1994.

[18] F.L. Lian, James R. Moyne, and Dawn M. Tilbury. Performance evaluation of control networks: ethernet, controlnet, and devicenet. *IEEE Control Systems Magazine*, 21:66–93, Fev 2001.

[19] F.L. Lian, James R. Moyne, and Dawn M. Tilbury. Network design consideration for distributed control systems. *IEEE Trans. On Control Systems Technology*, 10:297–307, Mar 2002.

[20] B. Lincoln. Dynamic programming and time-varying delay systems. Phd, Department of Automatic Control, Lund Institute of Technology, Lund, 2003.

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

[22] Pau Martí, Josep M. Fuertes, Gerhard Fohler, and Krithi Ramamritham. Improving quality-of-control using flexible timing constraints: Metric and scheduling issues. In *RTSS '02: Proc. of the 23rd IEEE Real-Time Systems Symposium (RTSS'02)*, page 91, Washington, DC, USA, 2002. IEEE Computer Society.

[23] Pau Martí, Josep M. Fuertes, Krithi Ramamritham, and Gerhard Fohler. Jitter compensation for real-time control systems. In *RTSS '01: Proc. of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*, page 39, Washington, DC, USA, 2001. IEEE Computer Society.

[24] R. C. Nunes and I. Jansch-pôrto. Qos of timeout-based self-tuned failure detectors: the effects of the communication delay predictor and the safety margin. In *Int. Conf. On DSN*, Jul 2004.

[25] K. Ogata. *Modern Control engineering*. Prentice Hall, Englewood Cliffs, 2nd edition, 1990.

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

[27] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of 1993 IEEE Int. Conf. on Neural Networks*, volume 1, pages 586–591, San Francisco, California, mar-apr 1993. IEEE/INNS. U. Karlsruhe.

[28] M. M. D. Santos and F. Vasques. Aplicando margem de jitter no projeto de sistemas de controle via rede. In *Proc. of VII Brazilian Workshop of Real-Time Systems*, pages 65–68, Fortaleza, Brazil, May 2005.

[29] K. G. Shin and H. Kim. Derivation and application of hard deadlines for real-time control systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(6):1403–1413, 1992.

[30] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 4th edition, 2003.

[31] The Mathworks. *Simulink Reference:Simulation and Model-Based Design*. The Mathworks Inc., Nantick, USA, oct 2004.