Detectores de Defeitos Adaptáveis para Sistemas de Controle Distribuídos de Tempo-Real sobre Redes *Ethernet*

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

Programa de Pós-Graduação em Mecatrônica [†]
LaSiD - Laboratório de Sistemas Distribuídos
Universidade Federal da Bahia

{aliriosa|macedo}@ufba.br

Resumo. *O presente artigo propõe um detector de defeitos adaptável baseado em redes neurais artificiais para um Sistema de Controle Distribuído de Tempo-Real (SCD-TR) sobre uma rede Ethernet. As simulações realizadas consideram várias métricas de Qualidade de Serviço (QoS) para detectores de defeitos, explorados no campo dos sistemas distribuídos convencionais, e evidenciam as vantagens da referida proposta quando contraposta com as principais abordagens adaptativas existentes. Adicionalmente, constatou-se que a sobrecarga causada pelo uso de detectores de defeitos nesse contexto não prejudica a realização de algoritmos de controle convencionais, como PID.*

Abstract. *This paper proposes an adaptive failure detector based on artificial neural networks intended to real-time distributed control systems over an Ethernet network. Simulations carried out on a simple replicated control application, where conventional metrics of Quality of Service (QoS) for failures detectors have been considered, evidence 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

Para a implementação de aplicações críticas de controle distribuído sobre redes convencionais é essencial promover soluções de detecção de defeitos adaptáveis e os algoritmos de adaptação utilizados 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 [2, 1, 7]. Essas propostas tratam a detecção de defeitos para sistemas distribuídos convencionais, não considerando aplicações de controle distribuído. Os autores de [5] 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*. Todavia, em [5], não são utilizadas redes de comunicação convencionais como a *Ethernet/CSMA-CD*. Devido ao seu baixo custo e suas altas taxas nominais de transmissão, a *Ethernet* 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. Assim, este 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 *Ethernet*; contrapõe tal proposta à detectores adaptáveis com *QoS* para sistemas distribuídos convencionais; e demonstra a viabilidade do uso de tal abordagem em um *SCD-TR*.

A seção 2 apresenta as principais abordagens de detecção adaptáveis com *QoS* para sistemas distribuídos convencionais. As seções 3, 4 e 5 discutem a abordagem de detecção proposta, os experimentos realizados, as conclções e as propostas de trabalhos futuros.

2. Abordagens Adaptáveis de Detecção de Defeitos

As abordagens de detecção apresentadas, em sua avaliação, utilizam as métricas de *QoS* para detecção de defeitos propostas por [2]. Tais métricas avaliam a rapidez na detecção de falhas e capacidade do

[*]Bolsista da FAPESB.

[†]Promovido conjuntamente pelos Departamentos de Ciência da Computação e de Engenharia Mecânica

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 [2], 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 $A_{k+1} - A_k = \Delta^i + (d_{k+1} - d_k)$. 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 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 [6, 2]. Desta forma, o marco no tempo FP (*Freshness Point* [2]) para chegada do próximo *heartbeat* é definido por $FP_{k+1} = EA_{k+1} + \alpha_{k+1}$. A adaptabilidade do detector consiste em ajustar EA e α. Assim, 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 [6] 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 [1] propuseram um detector adaptativo implementado em duas camadas, onde a camada inferior prover o serviço básico de detecção baseado nas métricas de *QoS* propostos por [2], 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 inferiror é baseada na estimativa proposta por [2], onde $EA_{k+1} = \frac{1}{n}\left[A_k - A_{k-n-1} + (k+1).\Delta^i\right]$. Para as n mensagens iniciais utiliza-se: $U_{k+1} = \frac{A_k}{k+1} \cdot \frac{k.U_k}{k+1}$, 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 [6], 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. Proposta de Detecção para Sistema de Controle Distribuídos

Como modelo de sistema, 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* [8] e então envia a informação de controle para nd^{atd}, o qual ativa τ^{atd} para realizar a atuação sobre o objeto controlado. Assume-se a possibilidade de falha por *crash* de nd^{ctrl}, deste modo, este elemento é replicado, formando um esquema redundante para tolerar uma única falha. A réplica primária (nd_p^{ctrl}) tem por responsabilidade: receber as mensagens oriundas de nd^{sns}; executar um algoritmo que garanta a consistência de estado com a réplica secundária (nd_s^{ctrl}); e enviar a ação de controle à nd^{atd}. Em nd_s^{ctrl} existe um detector de defeitos embutido para verificar falhas em nd_p^{ctrl}. 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. O subsistema de comunicação garante a ordenação na entrega das mensagens e não permite a existência de partições de rede nem de perdas de mensagens.

A predição baseada em RNA faz uso de uma MLP [3] 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 de propagação resiliente [9] para a atualização dos pesos sinápticos de acordo com uma taxa de aprendizado Δ adaptativa. Nesse 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 [9]. 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*; e *Taxa de heartbeat*. 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 $EA_{k+1} = EA_k + \Omega_{k+1}$.

4. Experimentos Realizados

Os experimentos 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. As análises observam a *QoS* em ambientes com atraso moderado, onde a rede é utilizada apenas para transferir dados entre os dispositivos do sistema de controle. Esse ambiente tenta refletir o comportamento do detector em um sistema de controle distribuído tradicional. Os experimentos comparam o desempenho dos algoritmos de [6] e [1] com a abordagem proposta. As experimentações utilizam $\Delta^i = \{100ms, 200ms, 500ms\}$, $\alpha_0 = 0ms$ 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 [1] e [6] .

Dada a 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_{k+1}^{hb}. 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 dos experimentos utilizou-se o *TrueTime* versão 1.3 [4], um ambiente de simulação para sistemas de tempo-real distribuídos, disponível na ferramenta *Simulink/Matlab* [10]. 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 *Ethernet/CSMA-CD* de $10Mbps$ e configurou-se os dispositivos 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 $\tau^{atd} : \{esporadica, D = 1000ms\}$. O impacto na atuação do controle é medido observando-se os instantes de chegadas das mensagens de controle no atuador sem e com o detector no sistema. Logo, sendo t_k e t_k^{fd} os instantes de chegada da *k-éssima* mensagem de controle no atuador sem e com o detector de defeitos, calcula-se o atraso d^{atd} na atuação por: $d^{atd} = \frac{1}{N} \cdot \sum_{k=1}^{N} (\left| t_k^{fd} - t_k \right|)$.

Os resultados obtidos estão resumidos na tabela 1. Nessa tabela, $A_{k+1} - A_k$ representa o tempo entre *heartbeats* consecutivos que chegam em $node_s^{ctrl}$ e serve como referência para analisar o quão próximo as estimativas de EA_k realizadas pelos detectores estão do valor real observado por

$node_s^{ctrl}$. Nos resultados, observa-se que o algoritmo baseado em *RNA* possui um desempenho melhor que os demais em todos os parâmetros avaliados. O algoritmo 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, obtive 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* nos experimentos se dá por conta das estimativas produzidas durante o procedimento de inicialização do detector. Analisando d^{atd}, pode-se concluir que o mecanismo de detecção nesse ambiente não influência no desempenho do controle, uma vez que o atraso inserido na atuação é extramente pequeno (embora haja uma leve tendência de crescimento para valores pequenos de Δ^i).

Tabela 1: Resultado em milissegundos da avaliação dos detectores de defeitos

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}	d^{atd}
	mean	*std*									
100,00	100,00	0,00	Jacobson	100,11	0,50	857,14	1075,44	0,00	0,00	22	$2,87 * 10^{-13}$
			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	$1,50 * 10^{-13}$
			Bertier	321,42	432,82	-	-	-	-	1	
			RNA	200,00	0,00	-	-	-	-	0	
500,00	500,00	0,00	Jacobson	500,56	2,50	51500,00	66468,00	0,00	0,00	3	$6,79 * 10^{-14}$
			Bertier	804,15	1084,26	-	-	-	-	1	
			RNA	500,01	0,00	-	-	-	-	0	

5. Conclusões e Trabalhos Futuros

Este artigo apresentou uma proposta de detecção de defeitos, baseada em rede neural, para um sistema de controle distribuído. Adicionalmente demonstrou-se, através dos experimentos, as vantagens da utilização desta proposta em relação as propostas de detecção para sistemas distribuídos convencionais existentes na literatura. Os resultados obtidos demonstram que em um ambiente de tráfego moderado, a implementação do detector de defeitos não influência no desempenho do sistema de controle. Os algoritmos do sistema (*RNA*, analisador de tráfego, controle *PID* e detectores de defeitos) foram implementados usando linguagem de *scripts* do *MatLab*. Como trabalho futuro, pretende-se avaliar o impacto do detector de defeitos apresentado (e mecanismos de recuperação de falha) em sistemas de controle com multiplos sensores, controladores e atuadores.

References

[1] M. Bertier, O. Marin, and P. Sens. Implementation and performance evaluation of an adaptable failure detector. In *Proceedings Of The International Conference On Dependable Systems And Networks (Dsn'02)*, pages 354–363, Washington, Dc, Jun 2002.

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

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

[4] 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.

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

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

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

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

[9] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of 1993 IEEE International Conference on Neural Networks*, volume I, pages 586–591, San Francisco, California, March-April 1993. IEEE/INNS. U. Karlsruhe.

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