Avaliação de Desempenho Temporal da Comunicação de um Sistema de Controle Baseado em Ethernet

Eduardo Kochenborger Duarte[1], João Cesar Netto[1], João Ricardo Wagner de Moraes[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brasil

{eduardo.duarte,netto,joao.moraes}@inf.ufrgs.br

Abstract. *In some industrial applications, the response time and the determinism are essential, making it necessary for the devices used to have a good performance. There are various factors that can lead to a performance degradation in a controller, like problems in the implementation of the communication with the I/O modules, or scheduling problems. The task of determining and otimizing the bottleneck of a device can be quite complex. This work has the objective of studying the impact in the performance caused by the communication between Nexto series controllers and I/O modules, proposing a methodology to be used as a tool in the otimization process.*

Resumo. *Em algumas aplicações industriais, o tempo de resposta e o determinismo são essenciais, sendo necessário que os dispositivos utilizados possuam desempenho satisfatório. Existem diversos fatores que podem levar à degradação do desempenho de um controlador, como problemas de implementação na comunicação com os módulos de I/O, ou problemas de escalonamento. A tarefa de determinar e otimizar o gargalo de um dispositivo pode ser bastante complexa. Este trabalho tem o objetivo de estudar o impacto no desempenho causado pela comunicação entre controladores da série Nexto e módulos de I/O, propondo uma metodologia para ser utilizada como ferramenta no processo de otimização.*

1. Introdução

A automação de tarefas outrora executadas manualmente é recorrente na indústria e causa grande aumento de produtividade. Por um longo tempo, a automação existiu em escalas bastante diminutas, utilizando-se principalmente de dispositivos mecânicos. Com o surgimento e disseminação da utilização de computadores, a flexibilidade adquirida possibilitou a automatização de praticamente qualquer tarefa [Gupta e Arora 2009].

Um controlador lógico programável (CLP) é um dispositivo que captura informações do ambiente e produz uma reação baseada na forma como foi programado. As informações vêm de sensores, como sensores de temperatura ou pressão, por exemplo. Usando esses dados, o CLP produzirá uma saída de acordo com a lógica programada pelo usuário. Por exemplo, pode-se ativar um atuador que controle um ventilador uma vez que a temperatura suba além de um limiar aceitável.

Existem diversos tipos de aplicações para as quais os CLPs são adequados. Naturalmente, é necessário fazer um levantamento de requisitos específicos à aplicação antes de se avaliar qual equipamento é o mais apropriado (utilizando-se alguma métrica). O

CLP de mercado da série Nexto, disponível no Instituto de Informática, em função de suas características, não possui o desempenho necessário para aplicações com requisitos de desempenho muito altos, considerando o tempo de ciclo como métrica.

Cada tipo de CLP é desenvolvido com o objetivo de se adequar a uma determinada gama de aplicações. A série Nexto é orientada para alta disponibilidade com soluções de redundância e suporte a grandes quantidades de pontos de entrada e saída. Outros equipamentos podem ter mais recursos de hardware, ou serem desenvolvidos para aplicações mais específicas. Um exemplo disso é mostrado na tabela 1 [Série Nexto 2017], [Beckhoff Product Overview 2017], [Beckhoff HW Documentation 2017], [Motion Controllers 2017].

Tabela 1. Comparativo de CLPs

CLP	CLP de Mercado	Tempo de Ciclo	Quantidade de Pontos
Uso geral	Nexto	Ordem de 5 ms	Grande
Uso geral com processador mais robusto	CX	Ordem de 3 ms	Grande
Motion Control	Q-Series	< 1 ms	Muito Pequena

O que se deseja é poder utilizar os controladores Nexto em sistemas de controle com requisitos muito fortes de desempenho. Existem indícios que o desempenho pode ser melhorado através de otimizações na comunicação do CLP com os módulos de I/O. O objetivo deste trabalho é formular uma estratégia para determinar quais os principais pontos passíveis de melhorias, identificando se a comunicação está impactando o desempenho. Para isso, serão apresentados testes que verifiquem o estado da situação atual no que diz respeito a desempenho, bem como uma análise sobre as possíveis causas e soluções.

A seção 2 contextualiza o leitor no que diz respeito à comunicação em ambiente industrial. Na seção 3, são apresentados conceitos básicos sobre CLPs. Em seguida, na seção 4, são apresentadas características, métricas e requisitos para sistemas de tempo real. A seção 5 descreve alguns trabalhos desenvolvidos na área de avaliação de desempenho. O método proposto para avaliar o desempenho da comunicação dos controladores está descrito na seção 6. Por fim, na seção 7, são apresentadas as conclusões do trabalho.

2. Comunicação em Ambiente Industrial

A comunicação através de redes digitais está bem estabelecida em ambientes industriais. Existem diversos sistemas *fieldbus* que são usados em indústrias comerciais, como Modbus e PROFIBUS, por exemplo. No entanto, com a grande disseminação do uso de Ethernet em outras áreas, existem tendências que levam ao uso de protocolos baseados em Ethernet na área industrial. No contexto atual, há uma grande produção de produtos e componentes adaptados a essa tecnologia, diminuindo custos e aumentando o interesse em utilizá-la [Neumann 2007].

No ambiente industrial, existe um conjunto de restrições que exige características não apresentadas pelo Ethernet convencional. Por este motivo, não é possível utilizar essa tecnologia diretamente. Diversos protocolos baseados em Ethernet foram desenvolvidos,

como EtherNet/IP, PROFINET e EtherCAT [Neumann 2007]. No equipamento disponível, a comunicação do controlador com os módulos de I/O possui diversas finalidades (detalhado na seção 6). O protocolo utilizado é o EtherCAT, um protocolo de comunicação industrial de tempo real e alta performance.

O EtherCAT funciona em uma configuração mestre/escravo, onde o mestre envia comandos aos nodos escravos, que podem escrever e/ou ler os dados [OMRON Automation Pvt Ltd 2016]. Através desse mecanismo, a atualização de I/O acontece ciclicamente no tempo. O objetivo é maximizar a utilização do canal de comunicação e minimizar os tempos de resposta.

Os comandos são inseridos no segmento de dados de um telegrama EtherCAT. Um telegrama é composto por um ou mais endereços de escravos, dados e alguns bits de controle. Esses telegramas são inseridos como *payload* em um *frame* Ethernet. Cada um desses *frames* pode conter diversos telegramas EtherCAT.

Um *frame* enviado pelo mestre é repassado apenas para o primeiro nodo escravo, que o repassa ao próximo, e assim sucessivamente. A leitura ou inserção de dados se dá neste momento, quando o *frame* está sendo transmitido, ou seja, não ocorre uma pausa na transmissão: os dados são lidos/inseridos *on the fly*. Uma vez que o *frame* chegue no último nodo escravo, este o enviará de volta ao mestre, utilizando-se da comunicação *full-duplex* do Ethernet. Assim, o tempo de processamento de cada escravo é praticamente desconsiderável, fazendo com que essa solução apresente o determinismo desejado.

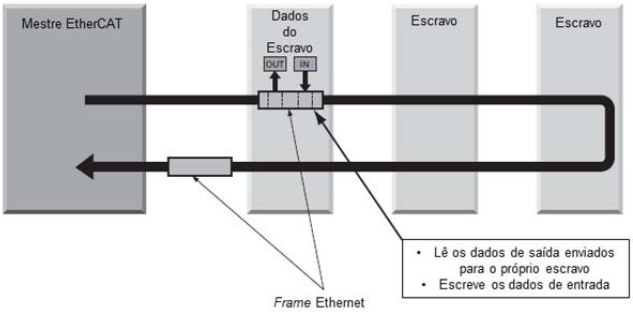

Figura 1. Comunicação EtherCAT (Fonte: Adaptado de [OMRON Automation Pvt Ltd 2016])

Embora o protocolo EtherCAT em si possua performance e determinismo ótimos, existem outros fatores externos ao protocolo que podem diminuir o desempenho. A meta deste trabalho é formular uma estratégia para determinar se a implementação atual da comunicação interfere na performance do equipamento.

3. Controladores Programáveis

CLPs são, em essência, computadores de propósito geral: recebem uma entrada, que é processada, e produzem uma saída, de acordo com sua programação. São otimizados para tarefas de controle, geralmente de tempo real, e para suportar ambientes industriais hostis. Isso inclui variações de temperatura, umidade, vibração mecânica, e ruído [Bolton 2015].

3.1. Entradas e Saídas

Um CLP precisa coletar informações do mundo externo para tomar decisões de controle. Essas informações chegam ao controlador através de suas entradas, que estão conectadas

a sensores. Similarmente, as decisões tomadas pelo sistema de controle chegam ao mundo exterior através dos módulos de saída, que são ligados a atuadores.

A decisão sobre usar I/O digital ou analógico depende da aplicação. Sinais discretos são adequadamente representados por entradas e saídas digitais, enquanto sinais contínuos são adequadamente representados por entradas e saídas analógicas. As entradas são lidas pelo CLP, que converte o valor lido para uma escala conveniente à aplicação.

A leitura de entradas, execução do programa e escrita de saídas são feitas de forma cíclica no tempo. O ciclo do CLP é apresentado na figura 2 [Bryan e Bryan 1997].

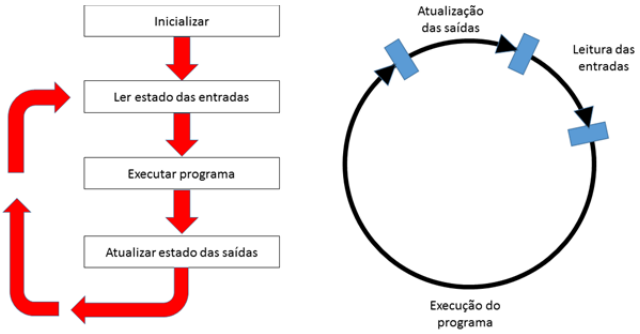

Figura 2. Ciclo do CLP

3.2. Execução: Tipos de Tarefas

Uma tarefa é um elemento de controle de execução capaz de invocar a execução de um conjunto de unidades de organização de programas (POUs), que incluem programas e blocos funcionais. A execução pode ser de forma periódica ou na ocorrência de eventos [IEC and others 1993]. Dessa forma, existem três tipos de tarefas:

- Tarefa Cíclica (*Cyclic Task*): executada em intervalos regulares de tempo. Este tipo de tarefa interromperá outras tarefas de prioridade mais baixa uma vez que tenha se esgotado o intervalo de tempo.
- Tarefa por Evento (*Event Task*): executada, apenas uma vez, na ocorrência da borda de subida da variável booleana especificada.
- Tarefa Contínua (*Continuous Task*): tarefa com execução contínua. O período de execução é definido pelo tempo de processamento gasto pela tarefa em um determinado momento. Portanto, o período de execução é variável, e será mais longo caso a tarefa precise de mais tempo de processamento.

4. Sistemas de Tempo Real e Determinismo

Um sistema de tempo real é definido como uma atividade ou sistema de processamento que precisa responder a entradas geradas externamente dentro de um período finito e especificado [Field-Richards 1983] (conforme citado em [Burns e Wellings 2001]). A dimensão deste tempo depende do tipo de aplicação.

No contexto de sistemas de tempo real, determinismo significa que o comportamento do sistema seja previsível em termos de valores e tempo [Kopetz 2011]. Em um sistema de frenagem, por exemplo, não é suficiente garantir que a ação de frenagem vai ser executada em algum momento posterior. Neste caso, seria necessário uma restrição mais forte, como definir que a ação ocorrerá 2 ms após o seu acionamento.

Assim, relacionando esses conceitos com os tipos de tarefas apresentados na seção 3.2, é possível perceber que tarefas contínuas não são adequadas para aplicações que necessitem de determinismo. Uma vez que não existe um tempo definido entre cada execução, não é possível saber antecipadamente em que momentos a tarefa executará.

4.1. Tempo de Ciclo

O tempo de ciclo é definido como o tempo entre a ativação da tarefa até o final da sua execução [Bolton 2015]. Isso inclui o tempo gasto pela atualização de I/O e o tempo de execução da lógica do usuário.

Os CLPs da série Nexto apresentam como tempo de intervalo mínimo entre ativações de 5 ms. Tanto aplicações como o sistema de frenagem mencionado na seção 4, quanto aplicações de *motion control*, precisam de tarefas que executem mais rapidamente.

Mesmo para uma tarefa que execute sempre o mesmo programa, existe uma variação neste tempo, chamada de *jitter* de tempo de ciclo. Atualmente, não é possível o uso desses CLPs nessas aplicações devido às características apresentadas pela série.

5. Trabalhos Relacionados

Sabendo-se que existem diversos componentes do sistema que podem ser otimizados em relação ao seu desempenho, o que se deseja é identificar qual deles é o gargalo. Neste trabalho foram usados equipamentos específicos. Por este motivo, não existem outros trabalhos especificamente relacionados com os mesmos dispositivos. Contudo, existem outros trabalhos de avaliação de desempenho que utilizam diferentes ambientes, ferramentas e equipamentos, ou mesmo trabalhos de cunho teórico.

Em [Jarp 2002], o autor propõe identificar gargalos em programas através do uso de *performance counters*. Com esse mecanismo de monitoração, são obtidas informações úteis sobre o programa em execução, como: quantidade de ciclos de processamento, número de instruções, ciclos de *stall* e atividade dos diversos níveis de cache. Essas informações são muito úteis para ajudar na identificação de gargalos.

Em [Jain 1990], são descritos critérios para a seleção de parâmetros para serem testados a fim de se identificar o gargalo do sistema. Parâmetros que não dizem respeito ao componente cujo desempenho se deseja melhorar podem ser omitidos, simplificando a análise. Também é descrito o uso de monitores, que são ferramentas utilizadas para observar atividades de uma determinada parte do sistema, coletando e analisando os dados.

6. Método Proposto

Sintetizando o que foi apresentado nas seções anteriores, o objetivo é tornar o tempo de ciclo mínimo menor, determinando se o desempenho poderia ser melhorado através de melhorias na implementação da comunicação do CLP com os módulos de I/O. Contudo, não podemos descartar a possibilidade de que a comunicação não seja o único ponto responsável pelo desempenho não satisfatório. O método proposto considera isso.

Primeiramente, deseja-se saber qual a ordem de grandeza do tempo de ciclo dada a implementação atual. Este é um passo inicial para avaliar se são necessárias otimizações e para estimar a magnitude das mudanças com base nas metas estabelecidas.

Caso seja detectada a necessidade de um melhor desempenho, pode-se prosseguir para identificar os gargalos. O método proposto é apresentado na figura 3 e tem o objetivo de servir como ferramenta para o processo de otimização. Os testes e as ações foram numerados na figura para auxiliar a compreensão.

Avaliar o desempenho sem comunicação (I): para determinar se o desempenho é de fato limitado pela comunicação, realiza-se um teste para avaliar o caso sem comunicação. Caso o desempenho não atinja a meta estabelecida, constata-se que a comunicação não é o principal gargalo. Sugere-se investigar funções de *callback* de outras funcionalidades do sistema que possam estar interrompendo a execução do programa. Além disso, sugere-se a utilização de um *profiler* para monitorar e analisar a execução. Se o desempenho for satisfatório, prosseguir para a avaliação (II).

Avaliar o desempenho com uma carga computacional constante (II): executar um programa de usuário sem comunicação com carga computacional constante, que ocupe uma parcela do tempo de execução da mesma ordem de grandeza do tempo ocupado pela comunicação. Observar se há *jitter* no tempo de ciclo. Com uma carga constante, idealmente o *jitter* seria pequeno. Caso o desempenho não seja aceitável, é possível que existam outras *threads* de usuário executando com prioridade maior causando interrupções no programa. Sugere-se investigar o escalonamento de tarefas, utilizando um *profiler* para determinar se a perda de desempenho é causada por problemas de prioridade. Se o desempenho for satisfatório, prosseguir para a avaliação (III).

Avaliar o desempenho com comunicação (III): se o desempenho for degradado apenas quando utilizado comunicação, significa que a comunicação é realmente um dos gargalos. Portanto, é importante tentar isolar qual das funcionalidades implementadas através de comunicação é o problema. Sugere-se uma abordagem que trabalhe com quatro diferentes partes da comunicação:

- Troca de I/O, responsável pela atualização de entradas e saídas e, portanto, de alta prioridade;
- Diagnósticos do CLP, de baixa prioridade;
- Comunicação relacionada ao *discovery cycle*, que é responsável pelo descobrimento de novos módulos;
- Interface de rede: possíveis problemas de gerenciamento de envio dos frames EtherCAT. Possível compartilhamento de recursos com a interface de rede que realiza a comunicação com o *software* de programação.

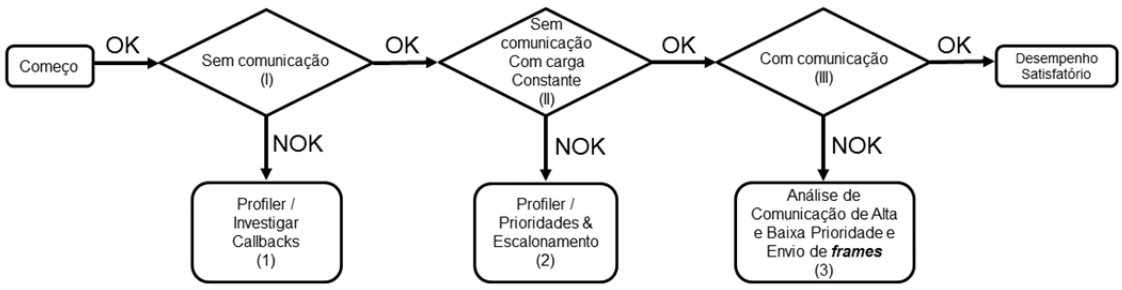

Figura 3. Fluxograma do método proposto

6.1. Resultados Preliminares

Conforme o método proposto, deseja-se avaliar a situação atual para determinar se mudanças são necessárias. Deseja-se fazer isso baseando-se em aplicações reais, utilizadas na prática, desses dispositivos. Assim, foi criada uma tarefa que não execute nenhum código de usuário, mas que realize comunicação. Deste modo, o tempo de ciclo da tarefa será composto apenas pelo tempo levado por outras funcionalidades, como descrito na seção 6, permitindo uma avaliação isolada da comunicação.

Como a meta do trabalho é possibilitar o uso da série Nexto em aplicações de alto desempenho, o intervalo de ativação foi definido como sendo o mínimo permitido pela ferramenta de programação, 5 ms, de modo a realizar a avaliação no contexto mais próximo dessas aplicações. As medidas de tempo de ciclo foram feitas segundo a definição apresentada na seção 4.1.

Foram realizados testes para determinar a quantidade de amostras que proporcionaria uma avaliação estável no que diz respeito à variabilidade dos tempos de ciclo. Assim, após variar o número de amostras, concluiu-se empiricamente que 150.000 amostras produziam um resultado estável, mostrado na tabela 2.

Tabela 2. Resultado do teste de tempo de ciclo.

Tempo de Ciclo Máximo (μs)	Tempo de Ciclo Mínimo (μs)	Tempo de Ciclo Médio (μs)
2686	1759	1918

Considerando que o tempo de ciclo máximo medido foi de cerca de 2.6 ms, é justificável que o intervalo de ativação mínimo da tarefa seja 5 ms. É preciso manter uma certa fatia de tempo para a execução do programa do usuário. Com esta configuração, no pior caso, tem-se aproximadamente metade do tempo de intervalo para execução de lógica. Tendo sido avaliada a situação atual e determinados os objetivos, o método proposto na seção 6 pode ser utilizado como ferramenta para identificar pontos a serem melhorados.

7. Conclusões

Neste artigo, foi feita uma revisão sobre assuntos relevantes ao estudo. Assim, foi consolidada a proposta, com a execução de testes e produção de resultados preliminares. A abordagem proposta neste trabalho tem como objetivo sistematizar uma metodologia que torne mais fácil a localização dos pontos a serem otimizados.

Conforme os resultados preliminares apresentados na seção 6.1, a série Nexto é passível de melhorias. Os testes realizados foram concebidos com o objetivo de observar o desempenho em uma situação similar às encontradas em aplicações reais. Através dos tempos de ciclo obtidos, com uso de parte da metodologia apresentada, foi constatada a necessidade de um melhor desempenho para o uso em determinados sistemas.

Os trabalhos futuros desenvolvidos a partir deste podem utilizar a metodologia para melhorar o desempenho, colocando-a em prática. Localizar a origem dos gargalos pode ser uma tarefa bastante complexa sem o uso de uma sistemática. A estratégia apresentada busca isolar diferentes componentes do sistema, tanto por serem de áreas diferentes, como comunicação e escalonamento, quanto isolar componentes dentro da mesma

área, como analisar a comunicação de alta e baixa prioridade. Assim, a análise é bastante simplificada. Apesar de o trabalho tratar de um estudo de caso, o método pode facilmente ser adaptado para outras implementações, tornando-o bastante versátil.

Referências

Beckhoff HW Documentation (2017). BECKHOFF CX9020 Hardware documentation. https://infosys.beckhoff.com/content/1033/cx9020_hw/9007202790037387.html. Acessado em 29/06/2017.

Beckhoff Product Overview (2017). BECKHOFF Fieldbus Components: Product Overview. https://infosys.beckhoff.com/english.php?content=../content/1033/tcsample_crestron/html/tckb_crestron_performance.htm. Acessado em 27/06/2017.

Beckmann, G. (2004). Ethercat communication specification, version 1.0. *EtherCAT technology group*.

Bolton, W. (2015). *Programmable logic controllers*. Newnes.

Bryan, L. A. e Bryan, E. A. (1997). *Programmable controllers: theory and implementation*. Industrial Text Company.

Burns, A. e Wellings, A. J. (2001). *Real-time systems and programming languages: Ada 95, real-time Java, and real-time POSIX*. Pearson Education.

Gupta, A. e Arora, S. (2009). *Industrial automation and robotics*. Laxmi Publications.

IEC and others (1993). Iec 61131-3. *Programmable Controllers-Part*, 3.

Jain, R. (1990). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons.

Jarp, S. (2002). A methodology for using the itanium 2 performance counters for bottleneck analysis. Technical report, Technical report, HP Labs.

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media.

Motion Controllers (2017). Motion controllers. http://dl.mitsubishielectric.com/dl/fa/document/catalog/ssc/l03014/qmotion.pdf. Acessado em 09/06/2017.

Neumann, P. (2007). Communication in industrial automationâĂŤwhat is going on? *Control Engineering Practice*, 15(11):1332–1347.

OMRON Automation Pvt Ltd (2016). Ethercat communication manual.

Série Nexto (2017). Série Nexto - Altus. www.altus.com.br/ftp/Public/Portugues/Produtos/Nexto/00DocSerie/ManuaiseApostilas/MU214000.pdf. Acessado em 29/06/2017.

Vosough, S. e Vosough, A. (2011). Plc and its applications. *International journal of multidisciplinary sciences and engineering*, 2(8).