PADRÃO DE ACESSO A DADOS OPC E SUA IMPLEMENTAÇÃO EM UM *DRIVER* OPC-MODBUS

Luiz Cláudio Andrade Souza[1]
Constantino Seixas Filho[2]
Ronaldo Tadêu Pena[3]

RESUMO

O controle de processos na indústria pode ser subdividido em três níveis distintos: nível de campo, onde a presença de dispositivos inteligentes integrados por redes *fieldbus* se torna cada vez mais comum; nível de processo, onde sistemas DCS e SCADA disponibilizam grandes quantidades de dados; e nível gerencial, onde tipicamente utilizam-se sistemas de bancos de dados e planilhas. A integração desses níveis, com o objetivo de se utilizar efetivamente toda a informação disponível em cada um deles, é de maior interesse para a indústria. A fim de possibilitar essa integração, o procedimento usual é o desenvolvimento de *drivers* de comunicação entre os diversos sistemas. No entanto, essa solução não é ideal pois demanda um grande esforço no sentido de se desenvolver *drivers* para uma grande variedade de equipamentos e de *softwares*, o que com o passar do tempo se torna de difícil manutenção, e portanto incapaz de acompanhar a evolução dos sistemas.

Na busca de uma solução para esse problema, um grupo várias de empresas de automação decidiu apoiar a criação de uma fundação não-lucrativa, cujo objetivo seria desenvolver um padrão de acesso a dados aberto, capaz de abreviar a necessidade de diversos *drivers* proprietários. A tecnologia OPC (*OLE for Process Control*) foi o resultado desse esforço. Além da premissa de garantir comunicação universal, o padrão OPC foi desenvolvido tendo em vista simplicidade de implementação e flexibilidade para atender às necessidades de diversos segmentos da indústria, agregando o máximo de funcionalidade sem comprometer sua eficiência. A infra-estrutura por trás do OPC é a tecnologia OLE, desenvolvida pela Microsoft inicialmente como modelo para comunicação entre aplicativos, mas que evoluiu a ponto de hoje estar intimamente envolvida nos próprios mecanismos dos sistemas operacionais baseados na plataforma Win32 (*Windows* NT, 95, 98). Além disso, a tecnologia COM (*Component Object Model*), fundação sobre a qual o padrão OLE foi implementado, vem sendo portada também para plataformas não-Microsoft. Assim, o padrão OPC se mostra uma alternativa robusta e altamente interoperavél. De fato, todos os maiores fornecedores de produtos para automação industrial atualmente oferecem ou estão desenvolvendo interfaces OPC.

Neste artigo pretende-se discutir a especificação OPC em si, bem como os aspectos relevantes das tecnologias OLE e COM, não apenas em termos conceituais mas também do ponto de vista prático, a partir de uma implementação de *driver* padrão OPC para um controlador lógico programável (Smar LC-700), utilizando sua interface serial e o protocolo Modbus-RTU.

[1] Engenheiro Eletricista, Mestrando em Engenharia Elétrica, PPGEE - EEUFMG.
[2] MSc., Professor Assistente do Departamento de Engenharia Eletrônica - EEUFMG, Diretor de Desenvolvimento da ATAN Sistemas de Automação.
[3] PhD., Professor Titular do Departamento de Engenharia Eletrônica - EEUFMG.

INTRODUÇÃO

O desenvolvimento do padrão OPC, *OLE for Process Control*, foi motivado pela constatação de que grandes quantidades de informação hoje disponíveis nos diversos níveis da indústria não são facilmente compartilhadas. Muitos esforços são dispendidos desenvolvendo-se *drivers* para a comunicação entre elementos díspares, porém essa solução é limitada, inconsistente e de difícil manutenção frente à rápida evolução de *softwares* e equipamentos. O objetivo fundamental da tecnologia OPC é prover uma infra-estrutura única, na qual a informação possa ser universalmente compartilhada. Além disso, as seguintes diretivas nortearam seu desenvolvimento:

- *Simplicidade de implementação*: o padrão é, à medida do possível, simples e pouco restritivo;
- *Flexibilidade*: há interesse em se endereçar as necessidades de vários segmentos da indústria;
- *Alta funcionalidade*: procura-se incluir o máximo de funcionalidade possível na especificação, sem conflito com os demais objetivos;
- *Operação eficiente*: embora a simples compatibilidade com o padrão OPC não garanta clientes ou servidores altamente eficientes, nada na especificação impede o desenvolvimento de *softwares* com essa característica.

O padrão OPC, conforme o próprio nome indica, é uma aplicação da tecnologia OLE tendo em vista as necessidades da indústria de controle de processos. Inicialmente, OLE era apenas um protocolo voltado à elaboração de documentos compostos, significando *Object Linking and Embedding*. Essa versão original foi construída sobre os mecanismos de DDE (*Dynamic Data Exchange*), e logo se mostrou insuficientemente robusta para seus fins. Quando uma segunda versão foi planejada, uma nova tecnologia de suporte, o COM (*Component Object Model*), foi também desenvolvida. A infra-estrutura provida pelo COM para a criação de componentes de *software* robustos se mostrou tão flexível que diversas outras tecnologias foram desenvolvidas seguindo esse modelo. Tais tecnologias receberam a denominação OLE, agora como um nome genérico. Apenas mais recentemente a denominação ActiveX assumiu esse papel, permitindo que o termo OLE voltasse a ter seu significado original.

Três tipos de acesso aos dados são definidos na especificação OPC: leitura e escrita síncronas, leitura e escrita assíncronas, e atualização enviada pelo servidor. Leitura e escrita síncronas, conforme o próprio nome indica, são executadas imediatamente pelo servidor, e só retornam para o cliente após completada a operação. Há dois tipos de acesso diferentes: ao *cache* normalmente mantido pelo servidor, ou diretamente ao dispositivo. Neste último modo as operações síncronas podem comprometer seriamente o desempenho do sistema, pois cliente e servidor ficam bloqueados enquanto o dispositivo físico é acessado. Operações assíncronas são mais eficientes, pois o cliente é imediatamente liberado após fazer a requisição, a qual o servidor pode processar da forma mais conveniente. Satisfeito o pedido, o servidor envia de volta ao cliente os resultados em uma única chamada de retorno. O terceiro tipo de acesso, baseado em mecanismos padrão OLE, permite ao cliente requisitar ao servidor que lhe envie, de forma periódica ou por exceções, mensagens atualizando um determinado conjunto de valores.

ARQUITETURA OPC

A arquitetura OPC pressupõe três objetos básicos: servidor, grupo e item (figura 1). Do ponto de vista do cliente, um servidor é essencialmente uma estrutura de armazenagem para grupos que, por sua vez, têm como função básica o armazenamento de itens. Esses itens, elementos mais simples na especificação, representam conexões a pontos de entrada ou saída. Assim, o item OPC não é um valor, mas apenas um meio de acesso a um valor. Desta forma, uma única variável de entrada ou saída pode

ser representada por itens diferentes, com propriedades distintas, e compartilhada por mais de um cliente.

É tarefa dos grupos reunir o conjunto de itens que interessam a um determinado cliente, assumindo o papel principal na interação cliente-servidor. Tal interação é feita através de interfaces, que reúnem grupos de funções relacionadas. São interfaces de um grupo OPC:

- *IOPCGroupStateMgt*: administração do grupo e de suas propriedades;
- *IOPCPublicGroupStateMgt**: complemento a *IOPCGroupStateMgt*, presente em servidores que suportam grupos públicos (interface opcional);
- *IOPCSyncIO*: funções de leitura e escrita síncronas;
- *IOPCAsyncIO*: funções de leitura e escrita assíncronas;
- *IOPCItemMgt*: adição, remoção e alteração de ítens;
- *IDataObject*: estabelecimento de conexões no sentido servidor-cliente.

Como a listagem acima indica, são os grupos os responsáveis por satisfazer pedidos de leitura e escrita, bem como por enviar atualizações para seus clientes, periodicamente ou por exceção. Essas transações de atualização podem ser ativadas ou desativadas a nível de grupo ou de itens individuais. Os grupos presentes em um servidor OPC são normalmente definidos pelos clientes, e somente o cliente criador do grupo pode acessá-lo; tal tipo de grupos é dito privado. Em alguns casos, porém, pode ser interessante que o servidor ofereça grupos passíveis de serem compartilhados por vários clientes. Quando essa capacidade é desejada, implementa-se a funcionalidade opcional dos grupos públicos.

As interfaces suportadas pelo objeto servidor são:

- *IOPCServer*: funções para criação e a remoção de grupos, bem como para acesso às propriedades globais de um servidor;
- *IOPCServerPublicGroups**: complemento a *IOPCServer*, em servidores que suportam grupos públicos (interface opcional);
- *IOPCBrowseServerAddressSpace**: interface que permite aos clientes navegarem pela estrutura de endereçamento do servidor, permitindo a criação automática de itens sintaticamente corretos (interface opcional);
- *IPersistFile**: permite aos clientes selecionar configurações de servidor armazenadas em disco (interface opcional).

Do ponto de vista do cliente, a função básica do servidor é prover uma infra-estrutura de suporte aos grupos. Além disso, cabe também a ele gerenciar aspectos relacionados à conexão com uma fonte de dados, tais como parâmetros de comunicação ou taxa máxima de amostragem. Embora essas propriedades sejam normalmente configuradas fora do ambiente OPC, o padrão oferece a possibilidade de se permitir aos clientes algum controle sobre as mesmas, através da interface opcional *IPersistFile*. Outra responsabilidade do servidor é implementar uma estrutura de endereçamento capaz de associar itens com variáveis reais. Apesar de a sintaxe específica com a qual um servidor realiza essa tarefa não ser definida no padrão OPC, existe a interface opcional *IBrowseServerAddressSpace* que permite aos clientes descobri-la dinamicamente.

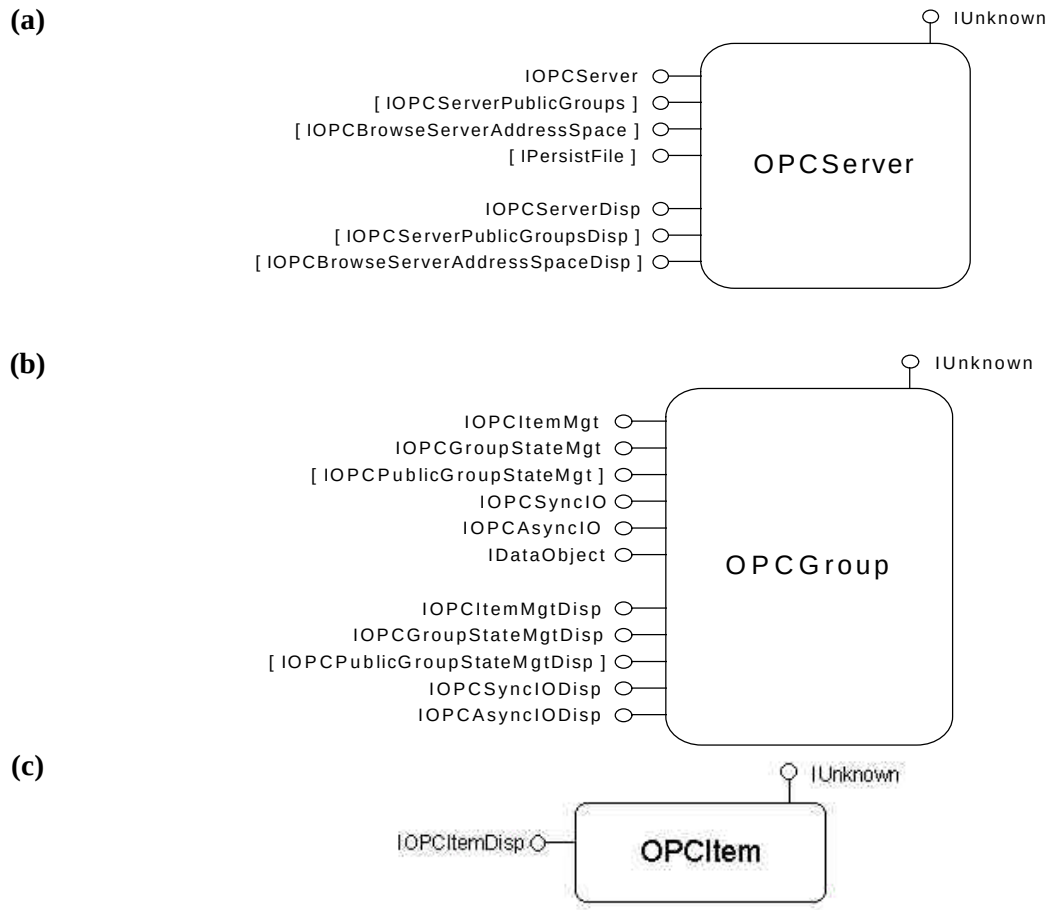

Figura 1: Objetos servidor (a), grupo (b) e item (c) OPC. (Interfaces opcionais entre colchetes).

Quanto à implementação, servidores OPC podem assumir a forma de *Dynamic-Link Libraries* (.DLL) ou de programas executáveis independentes (.EXE). Quando implementado sob a forma de DLL, ou *in-proc server*, o servidor executa no mesmo processo que o cliente, sendo essa configuração capaz de mais alto desempenho. Sob a forma de programa independente, porém, o servidor pode ser executado tanto na mesma máquina que os clientes (*local server*) quanto em outro nodo de rede (*remote server*). Esta última alternativa faz uso dos mecanismos DCOM (*Distributed COM*) que, quando apropriadamente configurados, tornam a infra-estrutura de rede completamente transparente para clientes e servidores. Por fim, há a possibilidade de construir servidores híbridos (*in-proc handlers*), implementados parte em DLL, parte como executáveis independentes, capazes de balancear desempenho e flexibilidade às custas de um desenvolvimento potencialmente mais complexo.

A implementação de clientes OPC é muito simples. O padrão define dois tipos de interface, *Custom* e *Automation*. Clientes escritos em linguagens compiladas como C ou C++ normalmente acessam o servidor através da interface *Custom*, de alto desempenho. A interface *Automation*, assim chamada devido à tecnologia *OLE Automation*, foi projetada para oferecer acesso a clientes escritos em linguagens interpretadas como Visual Basic, Delphi ou Java. Visto que o servidor realiza toda a tarefa de monitoramento, o cliente tem apenas que conectar-se ao servidor, criar um ou mais grupos com seus ítens de interesse, e esperar notificações através de uma implementação relativamente simples da interface *IAdviseSink*.

DESCRIÇÃO DO SERVIDOR OPC-MODBUS

O servidor OPC-Modbus mencionado no título deste trabalho está sendo desenvolvido para acessar um controlador lógico programável Smar LC-700, que faz parte de uma planta experimental fieldbus pertencente ao Laboratório de Ensino de Controle e Instrumentação da Escola de Engenharia da UFMG. Conforme o próprio nome indica, este servidor OPC se comunica com o CLP através do protocolo Modbus, via uma conexão serial.

A característica mais importante do servidor OPC-Modbus é sua natureza *multithreaded*. Tal arquitetura é de implementação complexa, devido não apenas à necessidade de se utilizar recursos como seções críticas e semáforos, mas também a algumas restrições impostas pela própria biblioteca OLE, principalmente no que diz respeito ao compartilhamento de interfaces por múltiplas *threads*. No entanto, a recompensa é significativa em termos do desempenho: tarefas diferentes mantêm o *cache* atualizado, ou completam operações assíncronas, por exemplo. Dessa forma, maximiza-se a disponibilidade do servidor para atender novos pedidos.

A linguagem de programação escolhida para o desenvolvimento do servidor OPC-Modbus foi C++. Embora componentes OLE e COM possam ser desenvolvidos em qualquer linguagem, C++ concilia alto desempenho e naturalidade para expressar componentes e interfaces. A estratégia adotada para construir os componentes foi a herança múltipla de várias classes, cada uma implementando uma única interface. Embora tal técnica seja mais trabalhosa que um desenvolvimento monolítico, os ganhos em termos de robustez e flexibilidade para se aperfeiçoar interfaces individualmente, ou para acrescentar novas interfaces, compensam plenamente.

Um último aspecto de destaque é o encapsulamento do protocolo Modbus-RTU em seu próprio componente *in-proc*. Além de isolar o servidor OPC de todo o código necessário para acessar o dispositivo físico e de ser reutilizável, o desempenho dessa DLL equivale àquele que seria obtido com o uso de funções locais. Essa implementação tem como particularidade o fato de ser segura para uso por clientes *multithreaded*, colaborando para a maior eficiência do servidor.

AVALIAÇÃO PRELIMINAR DE DESEMPENHO

Como o servidor OPC-Modbus ainda não se encontra totalmente desenvolvido, apenas uma avaliação preliminar de seu desempenho pôde ser apresentada neste trabalho. A fim de fazer essa avaliação, submeteu-se o servidor OPC-Modbus a um teste de leitura no qual seu desempenho foi comparado ao do OPC Sample Server. Esse último servidor, disponibilizado pela OPC Foundation, é uma implementação bastante completa do padrão OPC, embora limitada em termos de funcionalidade. Ao contrário de um servidor real, ele não possui código para acessar uma fonte de dados, nem tampouco para decodificar endereços de itens. Assim, o OPC Sample Server é uma boa referência apenas no que diz respeito à implementação das interfaces OPC. Como o servidor OPC-Modbus inclui ainda manutenção e atualização do *cache* (tarefa que permaneceu habilitada durante o teste, mesmo que trabalhando *off-line* com dados simulados), e também decodificação de endereçamento dos itens (ainda não acelerada pelo suporte a *blobs* - recurso opcional definido no padrão), os resultados obtidos foram possivelmente favoráveis ao OPC Sample Server de modo artificial.

O teste aplicado é em si muito simples: desejando simular uma condição de pior caso, conecta-se ao servidor um cliente que nada mais é que um *loop* executando operações de leitura síncrona à maior velocidade possível. Essa operação é uma das mais ineficientes definidas na especificação OPC, pois bloqueia o servidor. No entanto, por isso mesmo fornece uma boa medida de velocidade de transação.

Duas condições foram testadas: a leitura de grupos grandes, com uma centena de itens, e a leitura de apenas um item. No primeiro caso, o teste estressa o servidor principalmente pelo tamanho da transação. O segundo caso enfatiza tempo de resposta do servidor, dado o tamanho mínimo da

transação. Os dois servidores passaram pelos mesmos testes, sendo que o servidor OPC-Modbus foi avaliado nas configurações de leitura de variáveis em ponto flutuante e de leitura de variáveis discretas. O OPC Sample Server foi testado somente com variáveis em ponto flutuante, por ser capaz de simular apenas esse tipo de dado. Os testes foram repetidos dez vezes para cada servidor, alternadamente. Média e desvio padrão dos tempos gastos para completar cada execução dos testes foram calculados.

RESULTADOS

Antes de discutir os resultados dos testes, é importante salientar que, na especificação OPC, os valores fornecidos por um servidor são sempre acompanhados de informação de qualidade e de um *timestamp*. Assim, ao se comparar as taxas de dados mostradas a seguir com aquelas obtidas em tecnologias não-OPC, deve-se ter em mente que, para transferir a mesma quantidade de informação, outros sistemas teriam em princípio que recorrer a três operações de leitura, ao invés de apenas uma no padrão OPC.

Tabela 1: Primeiro teste - 10000 operações de leitura, 100 itens por operação.

	Tempo (s)	Ítens/s	Transações/s
OPC - Modbus, Ponto Flutuante	13,193 ± 0,031	75798	758
OPC - Modbus, Discreto	13,071 ± 0,032	76508	765
OPC Sample Server, Ponto Flutuante	11,983 ± 0,022	83450	835

Tabela 2: Segundo teste - 100000 operações de leitura, 1 item por operação.

	Tempo (s)	Ítens/s	Transações/s
OPC - Modbus, Ponto Flutuante	21,922 ± 0,087	4562	4562
OPC - Modbus, Discreto	21,936 ± 0,100	4559	4559
OPC Sample Server, Ponto Flutuante	23,753 ± 0,091	4210	4210

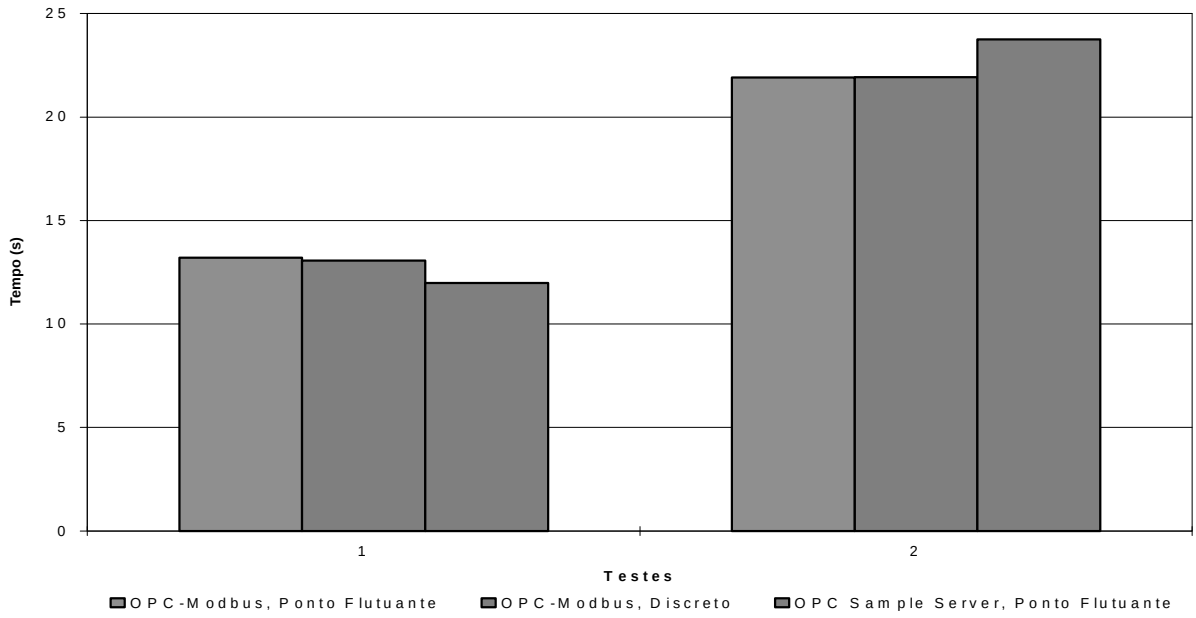

Figura 2: Tempo gasto pelos servidores OPC-Modbus (leitura de variáveis de ponto flutuante e discretas) e OPC Sample Server (apenas ponto flutuante) para completar os testes 1 (10000 operações de leitura, 100 itens por operação) e 2 (100000 operações de leitura, 1 item por operação).

DISCUSSÃO DOS RESULTADOS

Nas tabelas 1 e 2, e figura 2, observa-se que o servidor OPC-Modbus apresenta desempenho comparável ao do OPC Sample Server, a despeito de ser uma versão ainda em desenvolvimento e conseqüentemente não otimizada. Em particular, os resultados do segundo teste mostram uma menor degradação frente ao excesso de requisições, graças à opção de permitir acessos *multithreaded* pela biblioteca COM.

A diferença entre os testes, no que diz respeito a transações completadas e itens fornecidos por segundo, merece ser comentada. Por exemplo, entre o primeiro e o segundo teste o servidor OPC-Modbus com variáveis de ponto flutuante aumentou em 502% o número de transações por segundo. Ao mesmo tempo, o número de itens fornecidos por segundo caiu em 94,4%. Tais dados permitem afirmar que, à medida que o número de itens por transação diminui, o desempenho tende a aumentar até um ponto de saturação em que o número de transações por segundo atinge um valor máximo. Considerando-se que este teste representa um pior caso, e que em condições reais os clientes utilizarão mecanismos mais eficientes como atualização periódica, e que ainda esses mesmos clientes serão significativamente mais lentos (pois deverão processar os resultados das transações), é razoável propor que uma saturação do servidor dificilmente será alcançada na prática.

Outro ponto a ser comentado é a semelhança entre os resultados obtidos pelo servidor OPC-Modbus tanto para variáveis analógicas (ponto flutuante IEEE, 32 *bits*) quanto para as variáveis discretas. As operações de leitura e escrita em ambos os casos envolvem algum processamento por parte do servidor: no caso das variáveis discretas, cada grupo de oito delas é armazenado em um único *byte*; já as variáveis de ponto flutuante têm que passar por uma conversão de *big-endian* para *little-endian* (mudança na ordem em que os *bytes* que compõe a variável são armazenados na memória). No entanto, esses dois tipos de dados são transferidos entre cliente e servidor em estruturas do tipo VARIANT. Tais estruturas são utilizadas na tecnologia *OLE Automation* para armazenar variáveis e ponteiros. Enquanto a transferência de VARIANTs armazenando ponteiros envolve também a transferência do bloco de memória apontado, para tipos simples apenas a memória ocupada

pela VARIANT tem que ser transferida. Assim, o custo para se ler ou escrever variáveis analógicas ou discretas é aproximadamente o mesmo.

CONCLUSÕES

- O padrão OPC é plenamente capaz de cumprir sua promessa, ou seja, substituir uma infinidade de sistemas proprietários e *drivers* incompatíveis por clientes e servidores capazes compartilhar dados universalmente.

- Por ter sido desenvolvido com base em tecnologias estratégicas como OLE e COM, o padrão OPC se beneficia de quaisquer aperfeiçoamentos nessa infra-estrutura. Ainda devido às características inerentes a essas tecnologias, o padrão OPC é capaz de evoluir sem perder a compatibilidade com versões anteriores.

- A tecnologia OPC não é incompatível com alto desempenho. Os resultados dos testes, por exemplo, mostram um desempenho bastante razoável do servidor OPC-Modbus, mesmo que este ainda não inclua diversas otimizações previstas.

- À medida que o padrão OPC for sendo mais amplamente adotado, os diversos setores da indústria poderão concentrar seus melhores recursos e esforços em desenvolver soluções OPC altamente eficientes para as mais diversas aplicações.

REFERÊNCIAS

Brockschmidt, K., “Inside OLE, Second Edition”, Microsoft Press, 1995.

Chisholm, A., “DCOM, OPC and Performance Issues” White Paper, Intellution Inc., 1998.

OPC Foundation, “OLE for Process Control Data Access Standard, Version 1.0A”, OPC Foundation, 1997.

Rogerson, D., “Inside COM”, Microsoft Press, 1997.