# UMA METODOLOGIA DE INTERCONEXÃO ENTRE REDES DE CAMPO E DE SUPERVISÃO EM AMBIENTE INDUSTRIAL

Rodrigo Barbosa de Souza*, Adelardo A. Dantas de Medeiros*

*Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computação e Automação
Campus Universitário, CEP 59072-970
Natal, RN

Emails: rbsouza@dca.ufrn.br, adelardo@dca.ufrn.br

**Abstract**— A common problem in the industrial supervisory systems is that the access to the information on a industrial proccess, generally, is dependent of the automatized processs hardware elements. Each controller manufacturer, for example, can use a distinct communication method with its equipment. This paper considers the elaboration of a communication methodology that allows to access the process information in way that the communication strategy used in the industrial network does not intervene with the supervision system project. In this way, an interconnection system is considered that makes possible the execution of this task and guarantees a efficient communication and data changes. To evaluate the considered system, the proposed methology was applied in a oil artificial elevation automatized system.

**Keywords**— Supervisory Systems, Oil Well Automation.

**Resumo**— Um problema comum entre nos sistemas supervisórios atualmente utilizados na indústria se deve ao fato de que o método de acesso às informações sobre um processo industrial, geralmente, é dependente dos elementos de *hardware* que compõem a automação desse processo. Cada fabricante de um controlador, por exemplo, pode utilizar uma forma distinta de comunicação com seus equipamentos. Este trabalho propõe a elaboração de uma metodologia que permita acessar as informações do processo de forma que a estratégia de comunicação utilizada na rede industrial não interfira no projeto do sistema de supervisão. Para tanto, propõe-se um sistema de interconexão que viabilize a execução desta tarefa e garanta a eficiência na comunicação e troca de dados. Afim de avaliar o sistema proposto, aplicou-se a metodologia elaborada a um sistema automatizado de elevação artificial de petróleo.

**Palavras-chave**— Sistemas Supervisórios, Automação de Poços de Petróleo.

## 1 Introdução

Uma das grandes vantagens na automação de processo industriais está na possibilidade de utilizar controladores e dispositivos digitais com capacidade de processamento autônomo juntamente com PCs (Computadores Pessoais) (Cena and Valenzano, 2002). Assim, através de um sistema de supervisão, é possível obter informações sobre o processo físico controlado. Para tanto, utiliza-se um meio físico adequado para a transmissão de dados, criando um sistema de comunicação em rede em que os elementos podem trocar dados e compartilhar recursos entre si (Tait, 1998).

Um sistema de supervisão em um ambiente industrial automatizado é essencialmente composto por quatro elementos (Daneels and Salter, 1999):

**Processo Físico:** é o elemento principal do sistema e representa o objeto da supervisão;

***Hardware* de Controle:** é utilizado na interface física com o processo e, geralmente, no controle deste;

***Software* de Supervisão:** sistema computacional responsável pela aquisição, tratamento e distribuição dos dados; e

**Rede de Comunicação:** é responsável pelo tráfego das informações, constituindo-se, geralmente, de duas sub-redes denominadas *rede de campo* e *rede local de supervisão*.

A rede de campo é responsável pela aquisição dos dados do processo. No intuito de conseguir uma comunicação determinística, as redes de campo, em sua maioria, utilizam uma arquitetura *mestre/escravo*. Neste tipo de rede, os controladores que desempenham a função das estações escravas jamais iniciam a comunicação, respondendo somente às solicitações feitas pelo controlador mestre. Algumas das implementações mais comuns de redes que utilizam esta arquitetura são as redes *modbus* (Modicon Industrial Automation Systems, 2003) e *profibus* (PROFIBUS, 2002).

A rede local de supervisão, segundo Trung (Trung, 1995), é responsável por tornar disponíveis e compartilhar os dados do processo em uma *LAN* (Rede de Área Local) (Tanenbaum, 1997). Neste caso, os sistemas de supervisão, geralmente, utilizam arquiteturas do tipo *cliente/servidor* para acessar as informações do processo disponíveis na rede de campo, como observado por Giovanni Bucci (Bucci and Landi, 2003) e Liu Zhi (Zhi et al., 2000).

A disposição física dos elementos de um sistema supervisório pode ser observada na figura 1.

O *software* de supervisão é o principal responsável por tornar disponíveis os dados do processo. As formas de comunicação utilizadas pelos fabricantes do *hardware* de controle para acessar os dados do processo são, de um modo geral, muito vari-

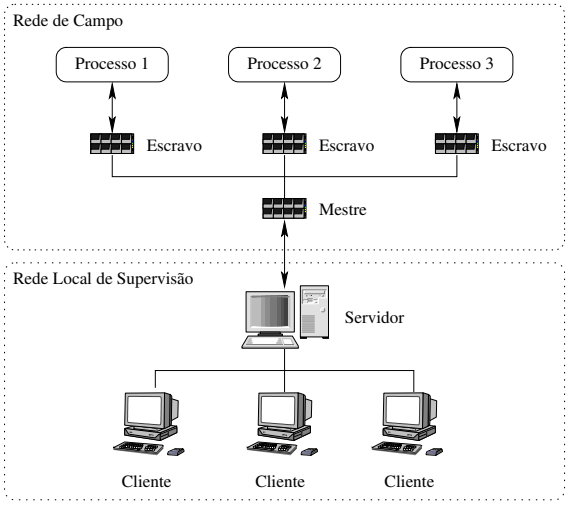


Figura 1: Estrutura Física de um Sistema Supervisório

adas. Com isto, é possível que para cada controlador utilizado nos processos seja necessário utilizar um *software* de supervisão distinto.

A elevação artificial de petróleo (Thomas, 2001) representa um caso típico de processos automatizados pelos mais variados fabricantes de *hardware* de controle. Em um sistema desse tipo, diferentes métodos de elevação, como por exemplo o bombeio mecânico e a injeção de gás, caracterizam-se como processos físicos de naturezas distintas. As empresas de automação de processos são cada vez mais especializadas. Assim, é comum encontrar poços de petróleo automatizados com controladores de fabricantes distintos. É possível observar, portanto, uma grande diversidade de sistemas supervisórios operando com a mesma finalidade, elevando o custo para a indústria e tornando desnecessariamente redundante a supervisão do processo. Neste sentido, este trabalho apresenta uma metologia de comunicação que elimina o uso de sistemas de supervisão redundantes, garantindo o acesso às informações do processo de forma independente em relação ao *hardware* de controle utilizado.

## 2 Metodologia de Comunicação

A metodologia proposta apresenta um protocolo de comunicação, denominado *Protocolo de Comunicação Inter-redes* (PCI), capaz de integrar as redes de supervisão e campo. A solicitação ou envio de informações utilizando o PCI deve ser iniciada sempre através de requisições. As requisições são processadas pela estação que interliga as duas redes, o servidor. O destino de uma requisição será sempre o servidor, que deve processá-la e respondê-la.

No PCI, para cada requisição recebida pelo servidor, duas respostas correspondentes, sendo uma parcial e outra definitiva, devem ser enviadas em resposta. A resposta parcial, ou réplica intermediária (RI), é gerada pelo servidor sempre que recebe uma requisição. O servidor deve enviar uma RI para a origem da requisição, afim de informar se tem condições (*acknowledge*) ou não (*not acknowledge*) de processar a requisição recebida. O servidor deve ainda, após processar a requisição, enviar para o(s) cliente(s) apropriado(s) uma resposta definitiva (RD) contendo os dados solicitados ou a confirmação dos dados enviados na requisição. A RD encerra a troca de informações iniciada por uma requisição. Utilizando *timeouts* para o recebimento das RI's e RD's, medidos a partir do envio de uma requisição, os clientes devem validar a comunicação entre as redes de campo e supervisão, bem como o eficiente processamento das requisições pelo servidor. A figura 2 ilustra o fluxo de dados utilizando o PCI.

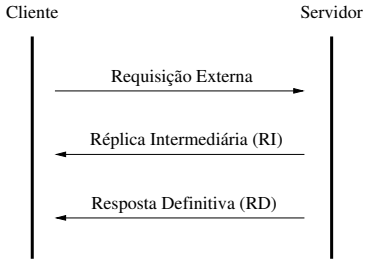


Figura 2: Fluxo de Dados PCI

A essência de uma requisição é a função lógica que ela representa e que deverá ser executada pelo servidor. As funções lógicas PCI classificam-se, quanto à finalidade, em:

- funções de controle; e
- funções utilitárias.

As funções de controle são aquelas utilizadas no gerenciamento da troca de dados utilizando o PCI. Essas funções têm origem, exclusivamente, no servidor e são utilizadas para informar aos clientes a chegada de requisições no servidor, erros ocorridos no processamento das requisições ou qualquer evento relevante que precise ser avisado aos clientes. Portanto, de acordo com o fluxo de dados observado na figura 2, as funções de controle se encontram somente em RI's e RD's, e nunca em requisições. A tabela 1 mostra algumas das funções de controle previstas pelo PCI:

| **Função** | **Descrição** |
|---|---|
| PCI_ERROR | Erro |
| PCI_ACK | Processará a requisição |
| PCI_NACK | Não processará a requisição |
| PC_WARN | Avisos |

Tabela 1: Funções de Controle

As funções utilitárias representam os serviços oferecidos pelo servidor e, portanto, devem ser utilizadas pelos clientes. As funções dessa classe são

definidas de acordo com os serviços exigidos do servidor. Assim, as funções utilitárias podem ser encontradas tanto nas requisições quanto nas réplicas.

As funções lógicas PCI subdividem-se ainda, quanto à forma de execução no servidor como:

- locais; e
- remotas.

As funções locais são aquelas executadas no servidor, sem que haja a necessidade de se comunicar com qualquer estação da rede de campo. As funções remotas exigem, obrigatoriamente, a comunicação com a rede de campo na sua execução.

## 3 Implementação da Metodologia

A implementação proposta busca utilizar uma forma simples, porém eficiente, de interconectar as redes de campo e supervisão através do protocolo PCI. Um sistema utilizando a implementação do PCI sugerida neste trabalho deve conter pelo menos uma aplicação do tipo servidor e uma ou mais aplicações clientes.

### *3.1 Clientes PCI*

Uma aplicação cliente deve ser capaz de enviar requisições ao servidor e tratar as respostas, quando recebidas, ou tratar a ocorrência dos *timeouts* quando não há nenhuma resposta para a requisição enviada. Para efetuar o controle de *timeouts* utiliza-se um *buffer* de requisições sem resposta, em que cada elemento no *buffer* deve indicar o estado da requisição e o momento do seu envio. O estado de uma requisição indica se o cliente já recebeu alguma resposta correspondente. A tabela 2 identifica os estados possíveis para as requisições no *buffer*.

| Estado | Descrição |
|---|---|
| PCI_WAIT_RI | Aguardando RI |
| PCI_WAIT_RD | Aguardando RD |

Tabela 2: Estados das Requisições

Na implementação dos clientes utilizam-se três processos paralelos. O primeiro processo, chamado de transmissor, aguarda solicitações dos usuários do sistema, gera uma requisição para essa solicitação, envia para o servidor e insere uma cópia da requisição no *buffer* de requisições sem resposta. O segundo processo, denominado processo receptor, aguarda a chegada de respostas, repassa o resultado ao usuário e retira sua requisição correspondente do *buffer*. O terceiro processo, chamado de processo de verificação, percorre continuamente o *buffer*, avaliando o tempo decorrido desde o envio de cada requisição. Além disso, esse processo deve verificar e modificar, se necessário, o estado das requisições, garantindo a aplicabilidade dos *timeouts* do PCI.

Note-se que o acesso paralelo dos três processos ao *buffer* de requisições sem resposta pode acarretar uma tentativa de acesso simultâneo. Para evitar esse problema, criou-se uma zona de exclusão mútua associada a um semáforo de *djkistra*, ou *mutex*, que gerencia o acesso a essa zona.

Afim de elucidar o funcionamento do sistema, representa-se na figura 3, utilizando uma *Rede de Petri*(Cardoso and Valette, 1997), o comportamento do processo transmissor, descrito nos seguintes passos:

**Passo 1:** processo deve ficar bloqueado aguardando solicitações do usuário.
**Passo 2:** monta uma requisição para a solicitação.
**Passo 3:** processo deve ficar bloqueado aguardando acesso à zona de exclusão mútua, chamada de ZEMC1, tentando decrementar o *mutex*, chamado de MC1.
**Passo 4:** insere cópia da requisição no estado PCI_WAIT_RI em BC1, juntamente com o momento do armazenamento.
**Passo 5:** incrementa MC1, liberando o acesso à ZEM1.
**Passo 6:** envia requisição para o servidor e **retorna ao Passo 1.**

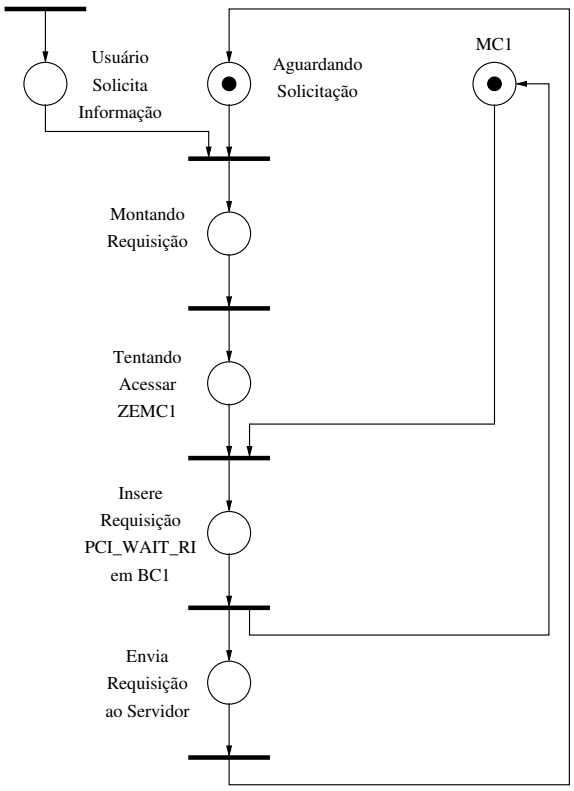


Figura 3: Processo Transmissor

Os processos receptor e de verificação podem ser modelados de forma similar.

### *3.2 Servidor PCI*

A aplicação do servidor precisa armazenar, processar e responder às requisições. Para realizar suas funções, o servidor utiliza cinco processos distintos e paralelos.

O primeiro processo, denominado de processo leitor, recebe as requisições, envia as RIs para os seus emissores e armazena as requisições. Se a requisição contém alguma função local, ela é armazenada em um *buffer* de requisições locais. Caso a requisição contenha alguma função remota, ela é armazenada em outro *buffer*, denominado *buffer* de requisições remotas.

O segundo processo, chamado de processo local, processa as requisições no *buffer* de requisições locais, retirando-as do *buffer*, executando a função solicitada ao servidor e armazenando a RD em um *buffer* de respostas.

O terceiro processo, chamado de processo remoto, processa as requisições no *buffer* de requisições remotas, retirando-as do *buffer*, comunicando-se com alguma estação da rede de campo e armazenando a RD no *buffer* de respostas.

O quarto processo, denominado processo escritor, retira as RDs do *buffer* de respostas e as envia aos clientes. Um quinto processo, chamado de processo interno, é utilizado na geração de alarmes e avisos.

A criação de *buffers* e processos distintos para armazenar e tratar as requisições que contêm funções locais e remotas de forma diferenciada se deve à diferença de velocidade do processamento inerente aos tipos de funções. Tratar estas requisições da mesma maneira introduziria um atraso desnecessário no tempo de resposta a requisições processadas rapidamente. Ao contrário das requisições, as RDs são processadas nos clientes. Portanto, no servidor, são tratadas da mesma forma, utilizando um único *buffer* para armazená-las e um único processo para enviá-las. Assim como no caso dos clientes, deve-se criar uma zona de exclusão mútua para cada recurso compartilhado entre os processos, bem como um *mutex* para cada zona criada.

Para evitar que os processos local, remoto e escritor permaneçam em espera ocupada enquanto não houver dados no *buffer* local, remoto ou de respostas, respectivamente, adotou-se a utilização de um semáforo bloqueante para cada *buffer*, afim de sinalizar a existência de requisições ou respostas em seus respectivos *buffers*. Para cada *buffer*, deve-se verificar se há pelo menos uma vaga disponível antes de inserir uma requisição ou resposta. Assim, utilizou-se um semáforo não-bloqueante para cada um deles, afim de sinalizar se ainda há espaço para inserções.

Os passos a seguir descrevem o funcionamento do processo remoto observado na figura 4:

**Passo 1:** processo fica bloqueado aguardando requisições no *buffer* remoto, denominado BS2.

**Passo 2:** processo fica bloqueado aguardando acesso à zona de exclusão mútua de BS2, ZEMS2, tentando decrementar o seu *mutex*, MS2.

**Passo 3:** retira requisição de BS2.

**Passo 4:** incrementa MS2, liberando o acesso à ZEMS2.

**Passo 5:** sinaliza vaga em BS2 incrementado o seu semáforo não-bloqueante, SNS2.

**Passo 6:** processa a requisição executando uma função remota e se comunicando com alguma estação na rede de campo.

**Passo 7:** verifica se há espaço no *buffer* de respostas, BS3, tentando decrementar o semáforo não-bloqueante de BS3, SNS3.

**Passo 8:** se não há espaço em BS3, **pára a aplicação servidora**, indicando falha no processo escritor.

**Passo 9:** se há espaço em BS3, processo bloqueia aguardando acesso à zona de exclusão mútua de BS3, ZEMS3, tentando decrementar o *mutex* MS3 de BS3.

**Passo 10:** coloca resultado da requisição em uma RD e insere em BS3.

**Passo 11:** incrementa MS3, liberando o acesso à ZEMS3.

**Passo 12:** O semáforo bloqueante SBS3 sinaliza resposta em BS3 e **retorna ao Passo 1**.

No Passo 8 é possível observar a ocorrência de uma situação anormal. O processo remoto é o responsável direto pela execução das funções remotas. Considerando que a atividade do processo escritor limita-se a retirar as respostas do *buffer* de respostas e enviar as RDs aos clientes, conclui-se que esse processo é bem mais rápido do que aquele. Assim, é conveniente supor que o *buffer* de respostas jamais deverá estar cheio, pois o fluxo de saída é bem maior que o de entrada. Desta forma, a situação de ausência de vagas nesse *buffer* deve ser tratada como uma exceção fatal no funcionamento do sistema.

Para representar esta situação, na figura 4 utiliza-se de um recurso que não faz parte do padrão de modelagem por Redes de *Petri*, o arco inibidor. Normalmente, para que uma transição seja disparada é necessário que haja pelo menos uma marca em todos os estados que condicionam o disparo dessa transição. O arco inbidor representa a operação complementar a esta. A existência de uma marca em qualquer estado que condicione o disparo de uma transição através de um arco inibidor impedirá o disparo da transição. As figuras 5 e 6 ilustram as condições de disparo de uma transição através de um arco inibidor.

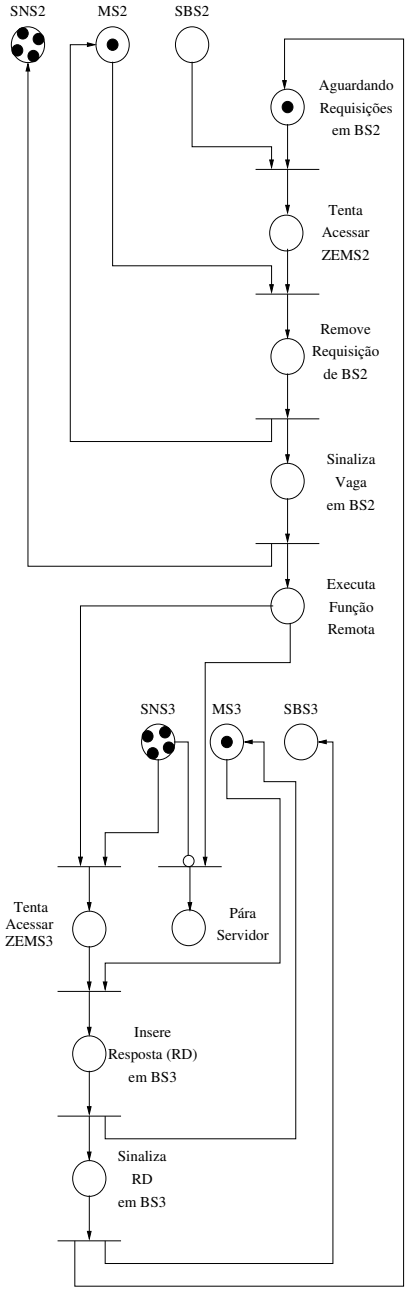


Figura 4: Processo Remoto

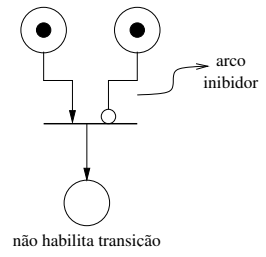


Figura 5: Transição Não Habilitada

## 4 Resultados Operacionais

A metodologia de comunicação sugerida neste trabalho foi implementada em um sistema de supervisão de poços de petróleo da indústria Petrobras na Unidade de Negócios UN-RN/CE. O sistema

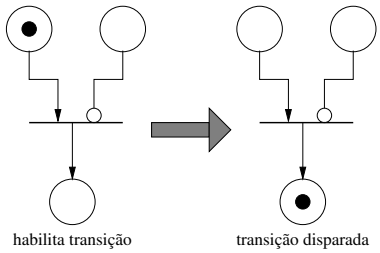


Figura 6: Transição Habilitada

foi testado na conexão de três clientes instalados em máquinas do escritório da empresa em Natal-RN conectados ao servidor localizado em Mossoró-RN, assim como os poços pilotos. Os elementos componentes do sistema de supervisão foram:

**Processo Físico:** a elevação artificial de petróleo através do método *gas-lift* (Thomas, 2001);

***Hardware* de Controle:** um controlador ZAP-500 da empresa HI Tecnologia.;

***Software* de Supervisão:** *Software* desenvolvido pela UFRN, em um projeto de parceria com a Petrobras;

**Rede de Campo:** meio físico utilizando um enlace de rádio com taxa de transmissão de 9600 bps e protocolo proprietário do fabricante do *hardware*; e

**Rede de Supervisão:** rede de comunicação privada da Petrobras.

Algumas das funções utilitárias do protocolo PCI criadas para adquirir os dados do processo foram as seguintes:

**PCI_READ_QGI:** função remota responsável por ler o valor atual da vazão de gás injetado no poço;

**PCI_RANGE_QGI:** função remota para configurar o instrumento de medição da vazão de gás injetado;

**PCI_CLOSE_PORT:** função local utilizada para forçar o fechamento do canal de comunicação com a rede de campo;

**PCI_SCAN:** função remota responsável pela captura e armazenamento dos dados do poço, afim de formar uma base de dados histórica.

A figura 7 representa uma tela do aplicativo cliente usada na calibração remota de um instrumento. O usuário deve preencher as informações de calibração no formulário e clicar sobre o botão *Calibrar*. Este procedimento gera uma requisição contendo a função PCI_RANGE_QGI, responsável pela configuração do controle do poço. O processo transmissor do cliente envia essa requisição ao servidor e aguarda uma resposta. Ao receber a requisição através do processo leitor, o servidor envia uma RI ao cliente indicando se está apto a processar a solicitação. Se a requisição for processada, por se tratar de uma função remota, o

processo remoto do servidor executará a calibração do instrumento, se comunicando com o poço através da rede de campo. O processo escritor enviará uma RD para o cliente com a confirmação da calibração ou uma indicação de falha na comunicação com a rede de campo. No cliente, esta resposta será tratada pelo processo receptor que deverá exibir o resultado ao usuário do aplicativo. Caso a resposta não chegue, o processo de verificação deve alertar o usuário a ocorrência de um *timeout*.

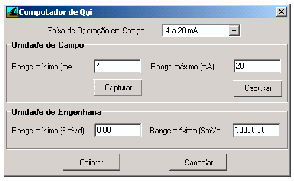

Figura 7: Aplicativo Cliente

## 5 Conclusões

A tarefa que a metodologia aqui apresentada se propõe a executar foi realizada com sucesso nos testes efetuaddos. O protocolo PCI permite que a sua implementação seja feita utilizando *sockets*, o que o torna flexível para ser utilizado em qualquer sistema operacional.

Um fator importante a ser considerado na utilização da metologia descrita neste trabalho é que somente o elemento de interconexão, o servidor, precisa conhecer os detalhes da comunicação com as estações escravas. Consequentemente, o acesso aos dados do processo pelo cliente será realizado de forma transparente e independente do *hardware* utilizado na rede de campo. No sistema de supervisão utilizado nos testes de implementação do padrão, e para qualquer sistema com características e elementos semelhantes a este, será possível agregar outros dispositivos de controle e comunicação com o processo simplesmente criando funções utilitárias que atendam às necessidades da indústria.

A adoção da metodologia proposta implica uma conversão de arquitetura mestre/escravo em cliente/servidor, apresentando como vantagens as seguintes características:

- Multi-Usuários: vários usuários do sistema poderão acessar quase simultaneamente um mesmo processo;
- Acesso Remoto: o acesso ao processo poderá ser feito remotamente por qualquer usuário com acesso à rede de supervisão;
- Multi-Clientes: os clientes poderão funcionar em interfaces distintas, como aplicações dedicadas ou *Web Browsers*; e
- Multi-Mestres: um único servidor é capaz de interconectar a rede de supervisão a várias redes de campo ao mesmo tempo através de seus mestres.

## Agradecimentos

O desenvolvimento do trabalho descrito neste artigo se deu através do projeto de automação de poços de petróleo financiado pelo CTPETRO e CENPES - Petrobras. Agradecemos o auxílio inestimável, sem o qual não seria possível a realização do projeto, dado pelos Engenheiros da Petrobras, em especial os Engenheiros Benno Waldemar Assmann, Edson Henrique Bolonhini e Luiz Sergio Sabóia Moura.

## Referências

Bucci, G. and Landi, C. (2003). A distributed measurement architecture for industrial applications, *IEEE Transactions on Instrumentation and Measurement* **52**(1).

Cardoso, J. and Valette, R. (1997). *Redes de Petri*, Editora da UFSC.

Cena, G. and Valenzano, A. (2002). A high performance field network based on a tree topology, *IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, pp. 105–109.

Daneels, A. and Salter, W. (1999). What is scada?, *International Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, Italy.

Modicon Industrial Automation Systems (2003). Modbus protocol reference, http://www.eecs.umich.edu/ modbus.

PROFIBUS (2002). Descrição técnica profibus 2002, http://www.profibus.org.br/.

Tait, A. (1998). Internets and intranets for industrial applications, *Hypermedia in Manufacturing Seminar*.

Tanenbaum, A. S. (1997). *Redes de Computadores*, Editora Campus.

Thomas, J. E. (2001). *Fundamentos de Engenharia de Petróleo*, Editora Interciência.

Trung, D. (1995). Modern scada systems for oil pipelines, *IEEE Petroleum and Chemical Industry Conference*, Denver, USA, pp. 299–305.

Zhi, L., Qin, J. S., Yu, T. B., Hu, Z. J. and Hao, Z. (2000). The study and realization of scada system in manufacturing enterprises, *IEEE World Congress on Intelligent Control and Automation*, Hefei, China.