ACIONAMENTO DE UM BRAÇO MECÂNICO UTILIZANDO A REDE MUNDIAL DE COMPUTADORES (WWW) ATRAVÉS DA LINGUAGEM DE PROGRAMAÇÃO JAVA E O PROTOCOLO RTP (*REAL TIME PROTOCOL)*

MARCOS ANTONIO ESTREMOTE, NOBUO OKI

Departamento de Engenharia Elétrica, FEIS, UNESP,

15385-000 - Ilha Solteira, SP, Brasil.

maestremote@aluno.feis.unesp.br, nobuo@dee.feis.unesp.br

Abstract— This work describes the development of a command Software for the control of a mechanical arm type SCORBOT ER - III. This software has the capacity to control and to monitor the robot remotely in real time through the World Wide Web (WWW), using libraries of JAVA as: native methods (JNI-JAVA Native Interface), Invocation of Remote Methods (RMI - Remote Method Invocation), Connecvity with database JAVA (JDBC - JAVA Database Connecvity) and JMF (JAVA Media Frameworks) with the Protocol of Real Time - RTP (Real Time Protocol). The robot control circuit was redesigned using the Altera Max Plus II environment and the connection between the robot and personal computer was made by the Parallel Port and digital cameras of the type WEBCAM, connected USB port.

Keywords— Remote control, Robot SCORBOT III - ER, JAVA Language, Real Time.

Resumo— Este trabalho descreve-se um Software de comando para o acionamento de um braço mecânico do robô SCORBOT ER - III. O software desenvolvido tem a capacidade de controlar e monitorar o robô remotamente em tempo real através da Rede Mundial de Computadores (WWW), utilizando bibliotecas de JAVA como: métodos nativos (JNI - JAVA Native Interface), Invocação de Métodos Remotos (RMI – Remote Method Invocation), Conectividade com Banco de Dados JAVA (JDBC – JAVA Database Connecvity) e JMF (JAVA Media Frameworks) com o Protocolo de Tempo Real – RTP (Real Time Protocol). Para controle do robô, o circuito de controle originalmente desenvolvido pelo fabricante, foi reprojetado utilizando-se o ambiente Max+Plus II da Altera e a conexão entre Robô e o PC é feita através de um dispositivo lógico programável tipo FPGA, que recebe os comandos provenientes da Porta Paralela do PC, o monitoramento através de câmeras digitais do tipo WEBCAM, conectados em uma Porta do tipo USB.

Palavras-chave— Controle Remoto, Robô SCORBOT ER – III, Linguagem JAVA, Tempo Real.

1 Introdução

De acordo com a literatura o primeiro trabalho envolvendo a manipulação remota de robôs via WWW foi efetuado no Projeto denominado Mercúrio coordenado por Ken Goldberg e Michel Mascha da Univesity Southern Califórnia (K. Goldberg, K. Mascha, M. Genter, N. Rothenberg, C. Sutter, and J. Wiegley, 1995). Atualmente, existem diversos grupos trabalhando com este conceito com finalidades que variam desde a educacional, visando despertar o interesse de alunos para áreas de computação, até o uso de sofisticadas teorias de controle, tais como a de redes neurais e técnicas de identificação para orientação de robôs em áreas remotas (manutenção de satélites, por exemplo) ou de risco (rastreamento de bombas e minas, prospecção e montagem de equipamentos para extração de petróleo em águas profundas).

Por outro lado, o uso da rede mundial de computadores, que se torna cada vez mais popular e atinge um público bastante heterogêneo, entre os quais, alunos que possam se interessar pela área de tecnologia. O desenvolvimento de projetos onde seja possível a interação de usuários com robôs parece ser uma grande oportunidade para despertar este interesse. A rede mundial de computadores possibilita, pela sua concepção, o acesso quase universal dos trabalhos desenvolvidos, pois a plataforma de uso encontra-se já definida e disponível. Na rede, existe uma série de páginas que efetuam este trabalho, no entanto o país carece de iniciativas neste sentido.

A área de robótica tem despertado interesse de pesquisadores pelo grande número de desafios que comporta, pois, sendo uma área de pesquisa multidisciplinar, necessita conhecimento das áreas de controle, engenharia mecânica, engenharia elétrica, cibernética, etc. Na área industrial os robôs já fazem parte das plantas industriais onde se faz necessário alta produtividade e confiabilidade, como a área automobilística.

Para o público leigo a área de robótica desperta fascínio pela divulgação, através da mídia, de recentes conquistas obtidas na área, como o envio de sondas tele-operadas a Marte, as aplicações na indústria automobilística, sua utilização em brinquedos e mesmo sua utilização em filmes de ficção.

Este artigo descreve o desenvolvimento de ambiente computacional que permita a manipulação de um braço mecânico com cinco graus de liberdade utilizando a rede mundial de computadores como plataforma de controle. Pretende-se com este trabalho disponibilizar uma página na rede mundial de computadores onde o usuário possa manipular objetos através de um braço mecânico com fins didáticos e edu-

cacionais, em contrapartida, comparar com implementações existentes, mostrando assim, algumas relevantes modificações em relação aos trabalhos anteriores.

2 Descrição

Para o desenvolvimento deste trabalho, foi utilizado um braço mecânico SCORBOT III – ER, disponível no Departamento de Engenharia Elétrica da Faculdade de Engenharia, Campus de Ilha Solteira – UNESP. Este braço possui cinco graus de liberdade, sendo controlado por seis motores de corrente contínua, possui também, juntas de rotação, caracterizando-o como um robô com articulação vertical. O mecanismo é acionado por *driver* elétrico, atua com garra tipo "dois dedos" e seu comando é feito ponto-a-ponto, como mostra a figura 1.

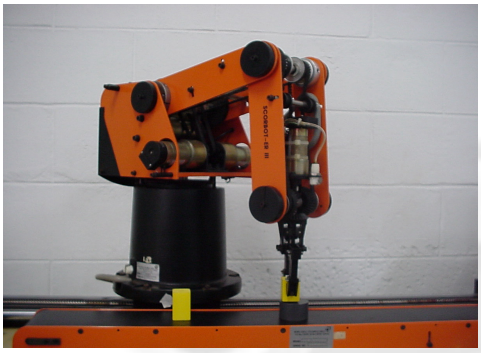

Figura 1: Braço Mecânico SCORBOT III – ER

Para controle do robô, o circuito de controle originalmente desenvolvido pelo fabricante, foi reprojetado utilizando-se o ambiente Max+Plus II da Altera e a conexão entre Robô e o PC é feita através de um dispositivo lógico programável tipo FPGA, que recebe os comandos provenientes da Porta Paralela do PC; Ao sistema estão acopladas câmeras digitais que permitem ao usuário verificar as posições e comportamento do braço mecânico controlado remotamente.

Trabalhos anteriores (B. Dalton, 2001) mostram a possibilidade de utilização de linguagem JAVA para este fim. JAVA, além de ser uma linguagem de programação, pode também ser utilizada como plataforma computacional (Sun Microsystems, 2003), que apresenta várias características, tais como, ser orientada a objetos, ser robusta e segura. Alguns ambientes necessários para elaboração deste artigo são descritos na literatura (H. Hu, L. Yu, P. W. Tsui, and Q. Zhou, 2001; A. Malinowski and B. Wilamowski, 2001; P. G. Backes, K. S. Tso and G. K. Tharp, 1999).

3 Implementação

Para o desenvolvimento do software de controle remoto do braço mecânico SCORBOT III – ER, utilizou-se a linguagem de programação Java versão 1.4.2 da *Sun Microsystems*, e alguns métodos e bibliotecas desta linguagem, como o uso de Invocação de Métodos Remotos (RMI – *Remote Methods Invocation*), Métodos Nativos (JNI – *Java Native Interface*) e JMF (Java Media Frameworks) para o monitoramento em tempo real do robô utilizando o protocolo RTP – Real Time Procol (Protocolo de Tempo Real). Nas figuras 2 e 3 são mostrados os layouts do software de controle.

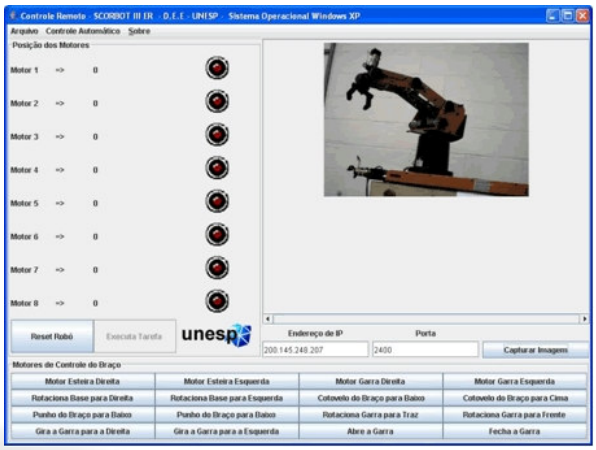

Figura 3: Layout Aplicativo Cliente

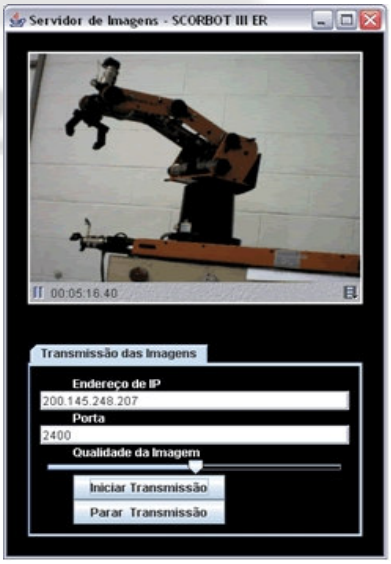

Figura 3: Layout Aplicativo Servidor

A seguir são detalhados os métodos para a criação do programa que têm por objetivo à Manipulação (acionamento) Remota do Braço Mecânico utilizando a Rede Mundial de Computadores (WWW) através da Linguagem de Programação JAVA.

3.1 Métodos Nativos

A plataforma JAVA proporciona o uso de códigos fonte escritos em outras linguagens. A biblioteca JNI nos auxilia na integração destas linguagens fazendo a

especificação de como é que se nomeia (dar nome aos métodos Java) e se invoca esses serviços de forma a que o *Java Virtual Machine* (JVM) possa localizá-los (Cornell, C. S. Horstmann, 2003).

Em muitas situações, já existe o código nativo implementado, não sendo necessário voltar a programá-lo. Assim, em vez de se programar, deve-se invocar estas funções já existentes a partir da lingaugem JAVA. Em alguns casos, o aplicativo exige acesso a recursos de sistema ou dispositivos que usando a tecnologia JAVA seria complicado, na melhor das hipóteses, ou impossível, na pior delas.

Para o acesso a Porta Paralela e controle do robô, utilizou-se um método nativo programado em C++, já que com a utilização da biblioteca JAVAX.COMM, específica da Linguagem Java para acesso a porta paralela, o controle torna-se insatisfatório, uma vez que, esta biblioteca não é portável a todas as plataformas computacionais existentes e possui erros em sua implementação (J. D. Ringgenberg, 2001).

3.2 Invocação de Métodos Remotos

Trabalhar com métodos remotos implica em possuir um conjunto de objetos colaborativos, que possam ser localizados em qualquer lugar. Esses objetos comunicam-se através de protocolos padrões de uma rede. Por exemplo, o objeto cliente envia uma mensagem para um objeto no servidor, contendo os parâmetros da solicitação. O objeto servidor reúne as informações solicitadas ou se necessário, comunica-se com objetos adicionais. Uma vez que o objeto servidor tenha a resposta para o cliente, este retorna as informações. O mecanismo RMI permite o acesso a um objeto em uma máquina diferente, podendo ser chamado de métodos de objeto remoto. Os parâmetros do método devem ser enviados para uma determinada máquina. Após a execução dos parâmetros no objeto remoto, este envia um valor de retorno para o objeto cliente (Cornell, C. S. Horstmann, 2003).

Na terminologia RMI, o objeto cujo método faz a chamada remota é denominado *cliente* e o objeto remoto é chamado de *servidor.* É importante lembrar que a terminologia cliente/servidor aplica-se apenas a uma chamada de método simples. O computador que está executando o código na linguagem de programação Java e chama o método remoto, é o cliente dessa chamada e o computador que contém o objeto que processa a chamada, é o servidor. Em qualquer momento, é possível que os papéis sejam invertidos.

Como mostra a figura 4, o cliente busca informações no servidor, através do método remoto *find,* este método retorna um objeto para o cliente.

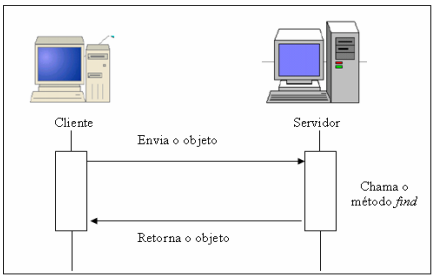

Figura 4: Solicitação do objeto remoto para o objeto servidor.

Executar o exemplo mais simples de objeto remoto exige muito mais configuração do que executar uma *applet* ou um programa independente. Deve-se executar programas nos computadores denominados servidor e cliente. As informações necessárias do objeto devem separar as interfaces do lado do cliente e as implementações no lado do servidor, no qual, através de pesquisas, o cliente localiza objetos no servidor.

No aplicativo de controle do braço mecânico, o cliente passa um parâmetro informando ao servidor qual motor deverá ser executado através de chamadas RMI. O servidor é encontrado uma vez que o mesmo possui uma nomenclatura que o diferencia dentro da rede. Depois de encontrado e de posse do parâmetro enviado pelo cliente, o servidor executa os comandos para a ativação dos motores fazendo uso de uma biblioteca dinâmica compilada em C++ e acessada através de terminações nativas – JNI.

3.3 JDBC – Conectividade com Banco de Dados JAVA

Esse pacote permite aos programadores se conectarem com um banco de dados, consultá-lo ou atualizá-lo, usando o SQL (*Structured Query Language,* acesso à banco de dados). JAVA e JDBC possuem uma vantagem fundamental sobre outros ambientes de programação de banco de dados, pois, os programas desenvolvidos com essa linguagem de programação são independentes de plataforma e fornecedor.

Acredita-se que futuramente devido à universalidade da linguagem de programação JAVA e do JDBC, estes finalmente substituirão as linguagens de banco de dados proprietárias.

Para o desenvolvimento deste projeto, utilizou-se JDBC para a conexão com um banco de dados Firebird.

Este banco de dados tem a finalidade de pré-programar tarefas a serem executadas pelo robô. A máquina cliente, que solicita a execução de uma tarefa de manipulação do robô, acessa remotamente o banco de dados no servidor e configura o deslocamento de cada motor pertencente ao braço mecânico com coordenadas específicas.

A conexão utilizando JDBC entre a aplicação e o banco de dados Firebird é feita informando o caminho de onde se encontra, na rede, o banco de dados a ser utilizado.

3.4 JMF (Java Media Frameworks)

A biblioteca Java Media Framework (JMF) é uma API que possui a capacidade de trabalhar com mídia baseada em tempo, como se faz em aplicações JAVA e applets, provendo apoio de captura e armazenamento de mídia. JMF utiliza o Protocolo de Transporte de Tempo Real (RTP) para receber e transmitir *streams* de mídia pela rede (*Java Media FrameWork API guide,* 1999).

O termo *Media Streaming* é usado freqüentemente para definir a técnica de envio e recebimento de pacotes em cima da rede em tempo real. Quando o conteúdo de mídia é transmitido a um cliente em tempo real, este pode começar a executá-lo sem ter que esperar seu carregamento completo – transmissões ao vivo. Quando o fluxo não possui uma duração definida, carrega-lo inteiro seria impossível.

Protocolos subjacentes diferentes de TCP são tipicamente usados para *Media Streaming*. Nesta linha, destaca-se o *User Protocol Datagram* (UDP). Este é um protocolo incerto, não se tem garantia que a informação alcançará seu destino e que chegará na ordem em que foi enviada. O receptor tem que compensar dados perdidos, pacotes duplicados e defeituosos.

O protocolo padrão da Internet para o transporte de dados em tempo real, como áudio e vídeo, é o RTP (*Real Time Protocol*).

O RTP pode ser usado em serviços de rede *unicast e multicast*. No serviço de rede *unicast*, são enviadas cópias separadas dos dados da fonte para cada destino. Já em um serviço de rede de *multicast*, os dados são enviados somente uma vez da fonte e a rede é responsável para transmitir os dados a locais múltiplos. *Multicasting* possui maior eficiência para muitas aplicações de multimídia, como vídeo conferências. O Protocolo padrão da Internet (IP – Internet Protocol) apóia *multicasting*.

Neste trabalho, o servidor captura imagens (*streams*) da manipulação do braço mecânico e as envia para um cliente específico na rede. Para o envio dos pacotes de informações, o servidor especifica para quem serão enviados os *frames* através de um endereço IP, e só a máquina cliente destino poderá ter acesso a estas informações – *unicasting*

Alguns autores, utilizaram para o envio e recebimento de pacotes de som e imagem, artifícios como a implementação de sub-programas escritos em linguagens como C++, Cobol, Fortran, etc; e acessadas utilizando o conceito de métodos nativos (JNI), esta forma de acesso e controle, acaba por diminuir a portabilidade dos aplicativos implementados, já outros, utilizam *script's* feitos em CGI (*Common Graphical Interface*), para fazer o monitoramento e controle do braço. Assim, o software implementado neste trabalho de mestrado, em relação à portabilidade, obteve uma crescente evolução em relação a trabalhos anteriores com a utilização de JMF/RTP.

4 Testes de Comunicação do Software

Neste tópico mostra-se o tráfego de pacotes na comunicação entre cliente e servidor na ativação dos motores do robô SCORBOT ER –III e também o tráfego de vídeo na rede na transmissão dos pacotes de dados para monitoração em tempo real. Estas informações foram coletadas na rede local da Faculdade de Engenharia de Ilha Solteira, onde houve a comunicação de máquinas cliente/servidor entre os computadores dos laboratórios de Circuitos Digitais (cliente) e o Laboratório de Controle (servidor).

4.1 – Tráfego de Pacotes na Rede

Como já mencionado anteriormente, a comunicação entre cliente e servidor é realizada de duas maneiras:

- ✓ Utilização de conceitos RMI, para a comunicação com o servidor remoto e;
- ✓ Conceitos de *media streaming* para envio dos quadros de som e imagem para a máquina participante na comunicação e controle do robô.

Devido ao alto índice de fluxo na rede em relação à transferência de pacotes, utilizando os protocolos TCP e UDP, procurou-se analisar os parâmetros na transmissão desses dados.

A figura 5, mostra um gráfico adquirido através do Gerenciador de Tarefas do sistema operacional *Windows XP*, onde é relatada a taxa de transferência de quadros de vídeos RTP na rede utilizada.

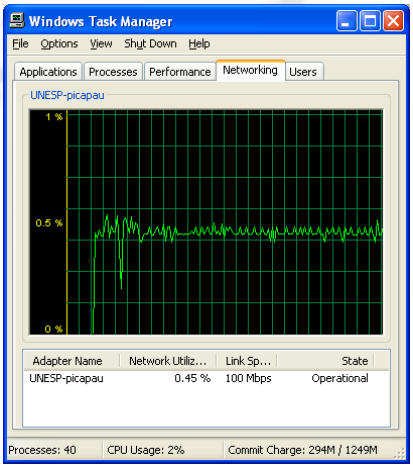

Figura 5: Utilização da Rede com Transmissão Vídea.

Como observado esta rede possui uma taxa de comunicação de 100 Mbps, e durante a execução da aplicação de controle na transmissão das imagens capturadas pelo servidor à máquina cliente, somente 0.45% desta banda foi utilizada. Em termos práticos, a rede operou com uma taxa real de 460,8 Kbps.

Já a figura 6, traz em seu bojo, informações coletadas a partir da execução do Gerenciador de Tarefas, quando a rede operava com a transmissão de pacotes de imagens (UDP) capturados pela webcam do servidor e enviadas ao cliente específico e, também pacotes utilizando conceitos TCP/RMI, pacotes enviados do cliente ao servidor com solicitação

do cliente ao servidor com solicitação de execução para movimentação de partes específicas do braço.

Verifica-se que a taxa de utilização do canal de comunicação de 100 Mbps, com o envio tanto de pacotes de vídeo RTP como de troca de mensagens RMI, ficou restringido a média de 0,83% de sua capacidade total, sendo assim, a taxa média de comunicação foi de 849,92 Kbps.

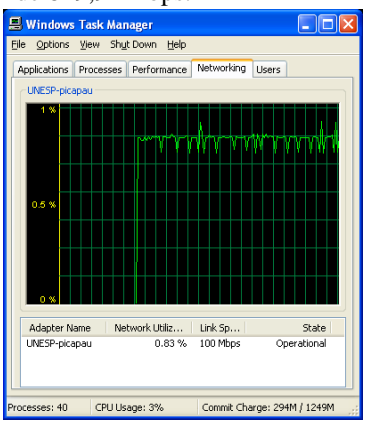

Figura 6: Utilização da Rede com Transmissão RTP e RMI

As análises esboçadas pelas figuras 5 e 6 foram melhores evidenciadas através de um software chamado *Ethereal – Network Protocol Analyzer* (*Analizador de Protocolos de Rede*), poderosa aplicação que faz análise do tráfego de dados da rede e reconhece centenas de protocolos. Esta ferramenta está disponível para as plataformas *Windows, Linux, BSD* e *Mac*.

4.2 – Análise dos Pacotes TCP/RMI Coletados na Rede

Com a ajuda do software de análise de tráfego (*Ethereal*), chegaram-se aos gráficos das figuras 7, 8 e 9.

A figura 7, apresenta as informações coletadas pelo software *Ethereal,* dos pacotes TCP/RMI na comunicação cliente/servidor. O tempo de coleta das informações foi de 123,281 segundos (00:02:03). Neste intervalo, foram coletados 2133 pacotes TCP/RMI perfazendo um total de 188277 bytes (183,63 Kbytes), tendo como tamanho médio dos pacotes transmitidos de 1,49 Kbytes/s.

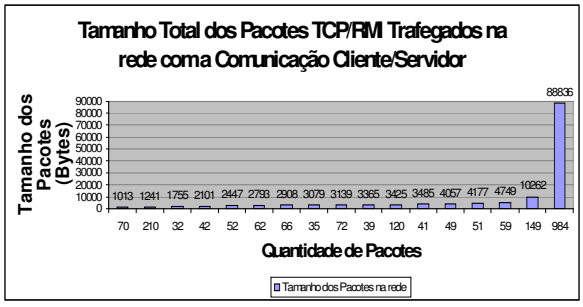

Figura 7: Pacotes TCP/RMI na comunicação cliente/servidor.

Na figura 8 apresenta a demanda de pacotes enviados pelo servidor à máquina cliente, e a quantidade de quadros enviados pelo cliente ao servidor solicitando a execução de alguma tarefa. Estas trocas de mensagens são oriundas do protocolo TCP, que é um serviço confiável. Com isso, há a necessidade na troca de informações, que o destino envie uma confirmação positiva ao destinatário para que a transmissão de novos pacotes se torne possível. Em caso de confirmação negativa o pacote é reenviado. Na figura 8, "A" representa o cliente e "B" o servidor. Nota-se que foram enviados 1158 pacotes ao servidor e este enviou um total de 975 pacotes ao cliente.

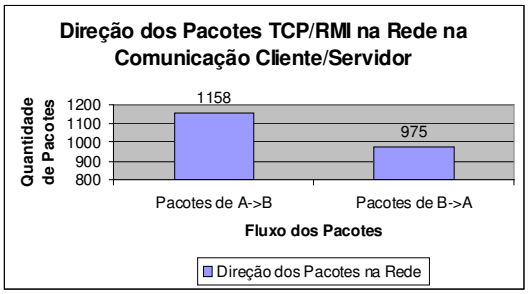

Figura 8: Direção dos Pacotes TCP/RMI

Na figura 9 é ilustrado o tamanho total em *bytes* dos pacotes enviados do cliente "A" ao servidor "B", e retornando estas informações do servidor "B" ao cliente "A".

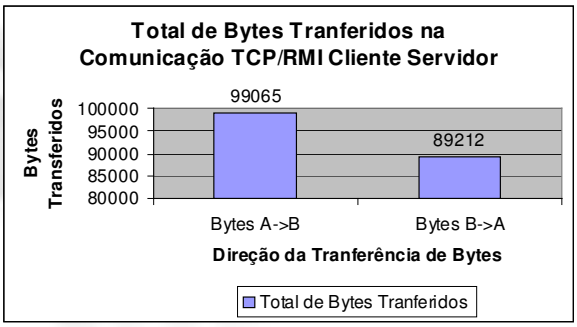

Figura 9: Total de *bytes* transferidos na comunicação TCP/RMI.

4.3 – Análise dos Pacotes UDP/RTP Coletados na Rede

A transmissão dos pacotes enviados pelo servidor à rede de computadores utilizando o protocolo UDP/RTP, é um serviço não confiável, pois, não garante a entrega dos pacotes enviados ao destino especificado. Os dados coletados na comunicação entre cliente/servidor na figura 10 apresentam um total de 13679 pacotes enviados, acumularam um total de 13042168 bytes transmitidos ou 12,42 Mbytes. A média de *bytes* transmitidos foi de 103,54 Kbytes/s.

4.4 – Análise do Fluxo de Pacotes dos Protocolos de Comunicação

A figura 11 apresenta o fluxo de dados obtidos na rede. Pode-se salientar que, a quantidade de dados do protocolo UDP/RTP foi mais expressiva por ser pacotes de conteúdo multimídia, e assim sendo, requer um canal de comunicação orientado a conexão. Já o protocolo TCP/RMI utiliza um canal de comunicação não orientado à conexão para a troca de men-

sagens. O protocolo TCP/RMI precisa ser invocado para que haja à troca de mensagens.

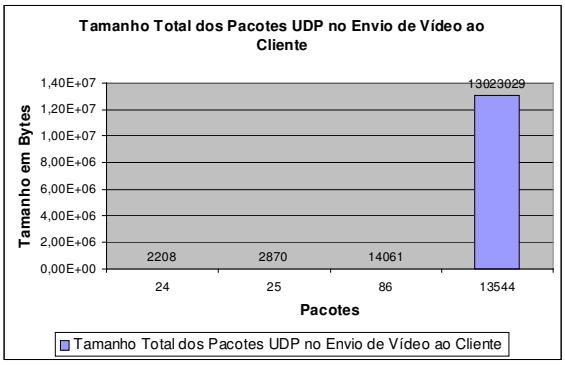

Figura 10: Tamanho dos Pacotes UDP.

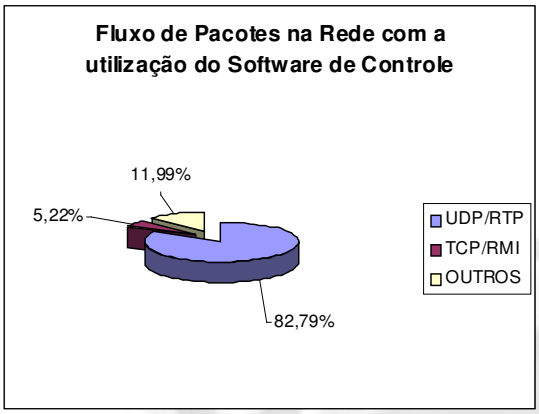

Figura 11: Fluxo de Pacotes na rede.

5 Conclusão

Os temas relacionados à comunicação remota de dispositivos eletrônicos devem ser vistos como um lugar em destaque, não apenas para pessoas que trabalham na área de automação, controle e informática, mas também para aquelas que, de alguma forma, utilizam a robótica como ferramenta para facilitar seus trabalhos.

O artigo nos mostra que foram empregados métodos disponíveis em Java, para fazer a comunicação de uma máquina cliente junto a um servidor, no qual ocorre à manipulação do braço mecânico via protocolo TCP/IP, além de monitorar os movimentos do braço remotamente através de uma WebCam Digital, utilizando o Protocolo de Tempo Real – RTP da biblioteca JMF.

Análises do software indicaram que a latência na comunicação para ativação dos motores do robô se manteve desprezível em uma rede local. No mesmo tipo de rede a transferência de quadros de imagens utilizando o protocolo RTP foi de boa qualidade, proporcionando ao cliente uma nítida visualização dos movimentos do robô remotamente. Já em uma conexão discada, a transferência de imagens se torna não confiável, pois, ocorre a perda de muitos pacotes na rede.

Os temas relacionados à comunicação remota de dispositivos eletrônicos devem ser vistos como um lugar em destaque, não apenas para pessoas que trabalham na área de automação, controle e informática, mas também para aquelas que, de alguma forma, utilizam a robótica como ferramenta para facilitar seus trabalhos.

Agradecimentos

Agradecemos, ao Departamento de Engenharia Elétrica de Ilha Solteira, ao Departamento de Pós-Graduação FEIS – UNESP, a CNPq pela ajuda financeira durante todo o projeto e a todos que contribuíram direta e indiretamente neste trabalho.

Referências Bibliográficas

A. Malinowski and B. Wilamowski. Controlling Robots via Intenet, In 1 st International Conference on Information Technology in Mechatronics, Istanbul, Turkey, October 1-3, pages 101-107, 2001.

B. Dalton, *Techniques for WEB Telerobotic*. Tese de Doutorado, Department of Mechanical and Materials Engineering University of Western Australia, Perth, Australia , 2001.

Cornell, C. S. Horstmann. *Core Java – Vol II - Recursos Avançados,* Editora – Makron Books, São Paulo, 2003.

H. Hu, L. Yu, P. W. Tsui, and Q. Zhou. Internet-based Robotic System for Teleoperation *In International Journal of Assembly Automation, Vol. 21, No. 2,* pages 143-152, 2001.

J*ava Media FrameWork API guide* – Sun Microsystems. (1999).

J. D. Ringgenberg. *Object-Oriented Design of Portable Control System Software.* Tese de Mestrado, Mechanical Engineering, University of California – Berkeley, 2001.

K. Goldberg, K. Mascha, M. Genter, N. Rothenberg, C. Sutter, and J. Wiegley. Desktop teleoperation via the world wide web. In *Proceedings of IEEE International Conference on Robotics and Automatio*n, 1995.

P. G. Backes, K. S. Tso and G. K. Tharp. The WEB Interface for Telescience, In *Presence, Vol. 8, No. 5, pages 531-539, October,* 1999.

Sun Microsystems. (2003). *The Java Tutorial.* Disponível na Internet via WWW. URL: http://java.sun.com/docs/books/tutorial/index.html.