APLICAÇÃO DE SOCKETS EM JAVA PARA MONITORAMENTO DE PROCESSOS EM ESTAÇÕES DE TRABALHO

Carla Estefanea de Castro Franco, Daniel da Silva
Sistemas de Informação - Associação Educacional Dom Bosco
Estrada Resende Riachuelo, nº 124. Campo da Aviação. Resende, RJ.
(carlaecfranco@superonda.com.br, daniel.dasilva@mpsa.com)

RESUMO

O objetivo deste artigo é demonstrar o conceito dos Sockets em JAVA e realizar uma abordagem na aplicação desta tecnologia em sistemas de monitoramento de estações de trabalho, confirmando assim, sua real importância e praticidade de aplicação.

INTRODUÇÃO

Atualmente, existem regras rigorosas dentro das corporações no que se refere à licença de software ou utilização de programas não homologados.

Uma organização pode ter funcionários que podem permanecer algum tempo com determinada licença de software ociosa em seus micros, sendo que outros podem precisar fazer uso desta licença. Em muitos casos, é inviável para uma empresa adquirir licenças de softwares específicos para todos os micro-computadores de sua rede. O ideal seria fazer um remanejamento das licenças ociosas.

Outro ponto importante é a falta de controle em relação aos softwares que não têm sua utilização permitida dentro da empresa. As conseqüências deste mau uso podem ser cobradas através de multas altas oferecidas pela fiscalização.

Fatores como estes levaram ao desenvolvimento de softwares de monitoramento das estações de trabalho. Os problemas acima citados poderiam ser prevenidos, pois através de tecnologias de comunicação em rede, as informações das estações poderiam ser transferidas para um servidor onde um administrador do sistema seria encarregado de tomar as devidas providências evitando, assim, maiores transtornos.

A seguir serão demonstrados os benefícios do uso desta ferramenta, Sockets em JAVA, no desenvolvimento de softwares com a finalidade de monitoramento. Será feito um estudo de caso baseado no software Sentinela, desenvolvido em conclusão do curso de Sistemas de Informação por Diogo Florenzano Avelino da Silva, José René Nery Cailleret Campanário, Luiz Filipe de Siqueira Hudson e Thales Schmalz Toniolo em 2003.

REDES DE COMPUTADORES E SUAS TECNOLOGIAS

As redes de computadores surgiram da necessidade de troca de informações e podem ser explicadas como sendo o conjunto de dois ou mais computadores interligados por meios eletrônicos e físicos, no intuito de trocarem informações de forma rápida e fácil, fornecendo aos utilizadores um campo mais vasto de atuação com compartilhamento de equipamentos, de recursos e de gerenciamento, tais como aplicações, ferramentas de comunicação, bases de dados, etc. A seguir, serão apresentados os conceitos necessários para um bom entendimento do tema abordado.

PARADIGMA CLIENTE/ SERVIDOR

O paradigma de programação distribuída através da separação das aplicações entre servidores (aplicações que disponibilizam algum serviço) e clientes (aplicações que usam esses serviços) foi a arquitetura de distribuição predominante nos anos 90.

Com este padrão de programação é acrescentada uma maior confiança ao sistema, pois a falha de uma máquina não compromete a operação do sistema como um todo e gera uma redução de custos, pois as máquinas clientes não precisam ser tão robustas, porque todo processamento é feito no servidor.

Geralmente, as aplicações clientes e servidoras são programas executados em máquinas distintas que trocam informações utilizando uma rede. Um cliente solicita determinado serviço e o servidor, por sua vez, provê algum serviço. Para uma aplicação cliente realizar alguma solicitação ela deve conhecer o endereço da aplicação servidora.

MODELO OSI

Com o objetivo de efetuar uma divisão das diversas partes da rede que compõem uma transmissão e para que possam existir etapas definidas e que permitam a integração dos diversos componentes, a ISO (International Standard Organization) criou o modelo OSI (Open Systems Interconections). Várias propostas de padronização foram geradas nas últimas décadas, porém o padrão OSI aponta como o modelo escolhido para sistemas de comunicação em nível de padronização mundial. Esse modelo serve de base para qualquer tipo de rede, seja de curta, média ou longa distância.

Este modelo ainda separa as etapas de transmissão, definindo como cada fase deve proceder na transferência dos dados. Isto torna flexível a implementação de softwares e hardwares ao longo da rede, pois define as funções de cada fase, facilitando a operacionalização para usuários e fabricantes.

Cada nível oferece serviços ao nível seguinte. As conexões de um nível são gerenciadas pelo protocolo daquele nível. Os níveis definidos como suas funções são sete assim enumerados:

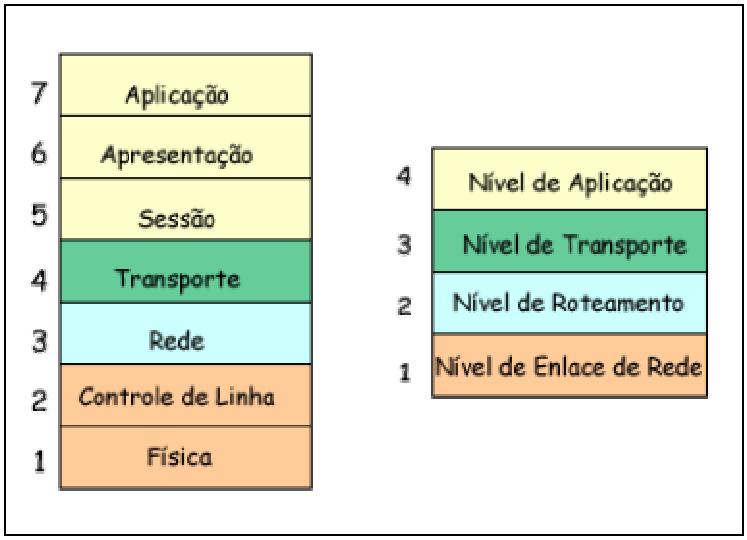

Fig. 1 - Níveis da Camada OSI

NÍVEL 7 – APLICAÇÃO: são os programas aplicativos do usuário, como: Banco de Dados, Transações On-Line, Correio Eletrônico, etc.

NÍVEL 6 – APRESENTAÇÃO: no qual ocorre a conversão dos dados, como, por exemplo: compressão de dados, conversão de formatos, conversão de códigos e criptografia, visando entregar dados à aplicação.

NÍVEL 5 – SESSÃO: é o que estabelece a conexão entre aplicações, definindo como serão feitas a troca de informações, o modo de transmissão, etc.

NÍVEL 4 – TRANSPORTE: controla a transferência de dados entre os computadores, garantindo a entrega da mensagem (bloco de dados) sem erros e na mesma ordem em que foi enviada. Tudo isso usando os dados fornecidos pelo nível anterior (nível de rede).

NÍVEL 3 – REDE: faz o encaminhamento de pacotes, contabilização e transferência de dados para outra rede. A unidade de transmissão é o pacote.

NÍVEL 2 – CONTROLE DE LINHA: faz a detecção e a correção de erros, fazendo com que a linha física pareça livre de erros. Organiza os bits do nível 1 em quadros.

NÍVEL 1 – FÍSICO: especifica as conexões elétricas, cabos, pinagem, nível de voltagem ou pulso de luz, sentido do fluxo de dados, etc. A unidade de transmissão é o Bit, representado pelos sinais elétricos.

PROTOCOLO TCP / IP

Para que exista comunicação em rede, faz-se necessário o uso de um protocolo de comunicação. Um protocolo funciona como "linguagem" entre computadores em rede e para validar esta comunicação, é preciso que os computadores utilizem o mesmo protocolo. Existem diversos tipos de protocolos. As aplicações de Sockets são feitas baseadas no protocolo TCP/IP.

O TCP / IP atualmente é o protocolo mais usado em redes locais. Isto se deve, basicamente, à popularização da Internet. Mesmo os sistemas operacionais de redes, que no passado só utilizavam o seu protocolo proprietário (como o Windows NT com seu NeTBEUI e o NetWare com o seu IPX / SPX), hoje suportam o protocolo TCP / IP. Uma das grandes vantagens do TCP / IP em relação aos outros protocolos existentes é que ele é roteável, isto é, foi criado pensando em redes grandes e de longa distância, onde pode haver vários caminhos para o dado atingir o computador receptor.

Outro fato que tornou o TCP / IP popular é que ele possui arquitetura aberta e qualquer fabricante pode adotar a sua própria versão de TCP / IP em seu sistema operacional, sem a necessidade de pagamentos de direitos autorais a ninguém. Com isso, todos os fabricantes de sistemas operacionais adotaram o TCP / IP, transformando-o em um protocolo universal, possibilitando que todos os sistemas possam comunicar-se entre si sem dificuldade.

Na transmissão de dados com o protocolo TCP/IP, ele divide as informações que serão transmitidas através da rede em "pacotes" de dados. Todo pacote que for enviado do servidor para o cliente ou vice-versa terá além dos dados que estão sendo enviados, uma série de dados de controle, tais como o número do pacote, código de validação dos dados e também o número da porta que o software utiliza. Quando o pacote chega em seu destinatário, o sistema lê no pacote o número da porta e sabe para qual aplicativo irá encaminhar o pacote. Esses controles são conferidos ao chegarem na outra ponta, pelo protocolo do receptor. Isso garante que um dado qualquer chegue a outro ponto da mesma forma que foi transmitido. Sendo assim, a integridade dos dados é mantida, independente do meio utilizado na transmissão, seja ele por linha telefônica, canal de dados, canais de voz, satélite ou qualquer outro meio de transmissão, já que o controle é feito nas pontas. Se na transmissão ocorre algum erro, o protocolo deve enviá-los novamente até que cheguem corretamente.

O TCP / IP é, na realidade, um conjunto de protocolos. Os mais conhecidos dão justamente o nome deste conjunto: TCP (Transmission Control Protocol, Protocolo de Controle da Transmissão) e IP (Internet Protocol), que operam nas camadas de Transporte e Internet, respectivamente.

A TECNOLOGIA DOS SOCKETS

A conexão por sockets tem origem em 1980, quando a ARPA (Advanced Research Projects Agency, Agência de Projetos de Pesquisa Avançados), órgão do governo norte americano, forneceu recursos financeiros para que a Universidade da Califórnia em Berkeley

oferecesse uma implementação UNIX do pacote de protocolos TCP/IP. O que foi desenvolvido ficou conhecido então como interface de sockets de Berkeley ou simplesmente sockets de Berkeley.

Sockets são estruturas que habilitam que dois ou mais aplicativos do tipo cliente / servidor que estão em rede, se conectem entre si. Ele representa um ponto de conexão para uma rede TCP / IP, muito semelhante aos soquetes elétricos encontrados em casas, que fornecem um ponto de conexão para os aparelhos eletrodomésticos.

Quando dois computadores querem manter uma "conversação", cada um deles utiliza um socket. Um computador é chamado servidor (ele abre um socket e presta atenção às conexões), o outro computador denomina-se cliente (ele chama o socket servidor para iniciar a conexão).

Para que seja efetuada a comunicação em rede através de sockets, é necessário ter conhecimento do número de IP ou HOST do computador e o número de porta do aplicativo ao qual se quer realizar a conexão.

O endereço IP identifica uma máquina específica na Internet e o número de porta é uma maneira de diferenciar os processos que estão sendo executados no mesmo computador. É exatamente a combinação do IP com o número de porta do aplicativo que se denomina Sockets.

Fig. 2 - Representação do endereço IP e o número de porta.

O protocolo TCP / IP oferece dois modos de utilização de Sockets: o modo orientado a conexão, que funciona sobre o protocolo TCP e o modo orientado a datagrama, que funciona sobre o protocolo UDP, que se localizam na camada de transporte do protocolo TCP / IP.

SOCKET TCP

O processo de comunicação entre aplicativos no modo orientado à conexão ocorre da seguinte forma: O servidor escolhe uma determinada porta e fica aguardando conexões desta porta. O cliente deve saber previamente qual a máquina servidora (HOST ou IP) e a porta que o servidor está aguardando conexões para solicitar uma conexão.

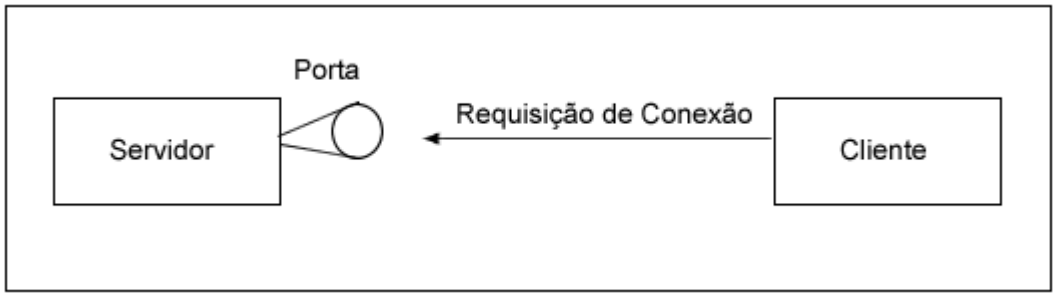

Fig. 3 – Requisição do Cliente

Se nenhum problema ocorrer, o servidor aceita a conexão gerando um socket em uma porta qualquer do lado servidor, criando assim um canal de comunicação entre o cliente e o servidor. A figura abaixo demonstra este canal de comunicação.

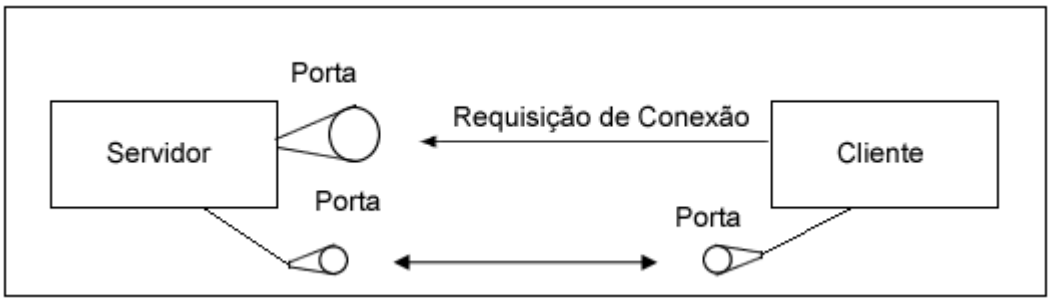

Fig. 4 – Criação do canal de comunicação com o Cliente

Tipicamente o comportamento do servidor é ficar em um loop aguardando novas conexões e gerando sockets para atender as solicitações de clientes. Os segmentos TCP são encapsulados e enviados em pacotes de dados.

Uma forma de visualizar o funcionamento de socket TCP seria compará-lo a uma ligação telefônica onde alguém faz uma ligação para outra pessoa e quando esta atende, é criado um canal de comunicação entre os dois falantes.

Um exemplo de aplicação de Sockets TCP são os mensageiros instantâneos como o ICQ e MSN, pois há necessidade de entrega garantida dos pacotes de dados.

SOCKET UDP

O User Datagram Protocol (UDP) é usado por alguns programas em vez de TCP para o transporte rápido de dados entre HOSTS dentro de uma rede TCP / IP. Em UDP não se estabelece conexão, pois a comunicação ocorre apenas com o envio de dados.

Neste tipo de Socket, uma mensagem é um datagrama, que é composto de um remetente, um destinatário ou receptor e a mensagem. Caso o destinatário não esteja aguardando uma mensagem, ela é perdida, pois este protocolo não realiza a verificação de transmissão de dados. A figura abaixo apresenta o envio de um datagrama de uma suposta máquina (Maq1) para outra (Maq2) em uma rede.

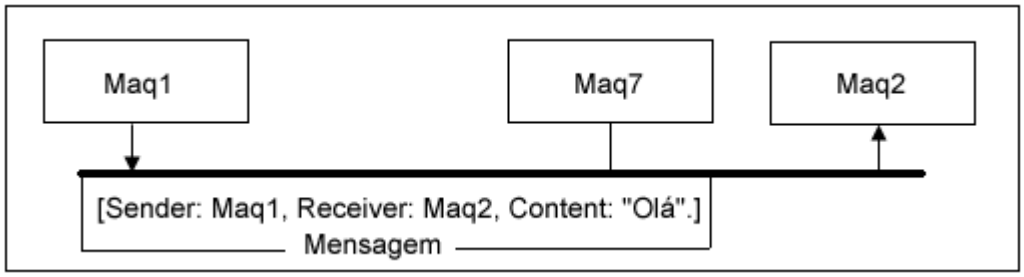

Fig. 5 – Envio de pacotes de dados

Fazendo uma analogia ao mundo real, os sockets UDP funcionam como um serviço de entrega de cartas onde se tem o remetente, destinatário e a mensagem dentro do envelope. Esta carta é entregue ao destinatário, porém não há garantia de quando ela chegou ou mesmo se ela chegou.

Os Sockets UDP são aplicados em aplicativos que priorizam a velocidade e que não exijam integridade dos dados. Um bom exemplo seria em jogos de rede.

O USO DE JAVA

Java foi uma linguagem projetava por um grupo de desenvolvedores onde o objetivo era criar uma linguagem de programação multiplataforma, que permitisse ao programador, gerar códigos para serem executados por quaisquer tipos de dispositivos, tais como PC, Mac ou até mesmo em PDA's e telefones celulares, independente do sistema operacional. Ela

depende somente que a JVM (Java Virtual Machine) esteja instalada no dispositivo. Com isso, Java tornou-se uma linguagem absolutamente portável, sendo esta sua principal vantagem.

Uma outra vantagem da utilização de Java para a programação em rede é que esta linguagem possui APIs (Application Program Interface) onde se tem funções encapsuladas, que facilitam o trabalho do programador. Por exemplo, para programação de sockets em C++ utilizaria-se as funções socket(), bind() e connect() para que o cliente conectar-se no Servidor, e em Java bastaria somente instanciar um
objeto da classe Socket passando o endereço e a porta do servidor: Socket s = new Socket("localhost", 2005). Para o uso da linguagem C++ utilizaria-se também
ponteiros, o que pode complicar a gerência de memória.

Como já foi visto, Java possui pacotes de classes que facilitam a programação. As classes para a programação de Sockets estão contidas no pacote Java.net. São elas:

- Socket – implementa um Socket cliente, utiliza TCP;
- ServerSocket – cria uma interface de rede necessária para que a aplicação possa funcionar como um servidor TCP, deixando um Socket servidor que fica "escutando" em determinada porta ligações dos clientes;
- DatagramSocket – cria um socket UDP, para clientes e servidores;
- DatagramPacket – empacota informação num pacote UDP com informações da porta e endereço IP do destinatário.

ESTUDO DE CASO

O Sistema Sentinela consiste em um sistema de monitoramento de processos, programas em execução, em estações de trabalho dentro de uma rede, seja ela corporativa ou acadêmica. Este deve permitir a uma corporação um maior controle de todo o uso de softwares e da rede. Suas principais funcionalidades são:

- Monitorar status da rede (verificação das máquinas que estão conectadas a rede);
- Monitorar usuários da rede (verificação dos processos que estão sendo executados em cada estação);
- Legalizar softwares (como o sistema faz uma verificação dos processos que são executados, pode-se ter um maior controle checando quais são os funcionários que utilizam determinada licença e qual sua freqüência de uso. Bem como, verificação do uso de softwares que não são licenciados pela empresa ou que não estão na lista de softwares permitidos.);
- Listar processos (através do comando ps é feito todo o controle de todos os processos que estão sendo executados nas estações conectadas à rede); e
- Aniquilar processos (fazendo o uso do comando kill, há ainda a possibilidade de finalizar a execução de algum processo).

Para o desenvolvimento do sistema foi feito uso de algumas ferramentas. São elas:

- Macromedia Flash MX: a utilização deste software permitiu a criação de interfaces atraentes e amigáveis para o usuário. Isso de forma rápida e fácil.
- Linguagem PHP: foi utilizada para tratar os dados recolhidos pela interface, tendo em vista que a versão do Flash que foi utilizada não dá suporte a banco de dados. O banco de dados utilizado foi o Mysql por ser gratuito e de fácil manipulação.
- JAVA: Para o desenvolvimento do cliente Sentinela foi utilizado JAVA, devido sua portabilidade tendo em vista que a idéia central era que ele monitorasse toda rede

independente do Sistema Operacional e facilidade de programação de aplicativos em rede.

Este software faz o uso dos Sockets TCP para comunicação entre o servidor Sentinela e os clientes. Este tipo de Socket foi adotado, pois era necessário ter a real perspectiva dos processos que estavam sendo executados nos clientes no momento da coleta dos dados. As informações que seriam trafegadas necessitam chegar na íntegra no servidor sem sofrer o risco de haver alguma perda de dados outro motivo do uso de Sockets no Sentinela foi porque todos os sistemas operacionais possuem implementação para esta tecnologia.

O que podemos ressaltar como desvantagem no uso da interface de Sockets é a segurança, pois a maneira mais fácil de se invadir um sistema é através de uma porta que está aberta, um Socket aberto. A solução para este problema seria o uso de criptografia, porém isto não foi aplicado no Sentinela, pois os dados trafegados na rede não eram sigilosos. Tratava-se apenas da lista de softwares que estavam sendo executados em determinado momento em alguma estação.

CONCLUSÃO

Atualmente, sistemas com a finalidade de monitoramento de processos em estações de trabalho é essencial dentro de uma corporação, tendo em vista que existem leis e multas altíssimas em relação ao licenciamento de softwares. Outro benefício que este sistema proporciona é o total gerenciamento de todas as atividades dos usuários que estão conectados na rede.

Poderiam ter sido utilizadas outras linguagens para comunicação entre o cliente e o servidor, como C++, porém o uso de Sockets em JAVA para sua implementação garante portabilidade, confiabilidade e simplicidade na sua implementação devido às classes prontas para este fim. Com base nestas afirmações, podemos concluir que Java é a tecnologia ideal para esta categoria de software tendo em vista que existe a necessidade que ele seja executado em estações independentemente de Hardware e SO.

REFERÊNCIAS BIBLIOGRÁFICAS

Chan, M.C., Griffith, S. W., Iasi, A.F., **JAVA 1001 DICAS DE PROGRAMAÇÃO**, Ed. Makron, RJ, 1999.

Sousa, L.B.S., **REDES DE COMPUTADORES – DADOS, VOZ E IMAGEM**, 6º edição, Ed. Erica, RJ, 2002.

Taylor, **TCP / IP**, Ed. InfoBook SA, 1996.

Hopson, K.C., Ingram, S. E., **DESENVOLVENDO APPLETS COM JAVA**, Ed. Campus, 1997.

Torres, G., **REDES DE COMPUTADORES – CURSO COMPLETO**, Ed. AXCEL, 2001.

Nunes, L.R., **SOCKETS EM JAVA**, http://www.sumersoft.com/publicacoes/SocketsEmJAVA.pdf. Acessado em: 6 de março de 2005.

Battisti, J., **INTRODUÇÃO AO TCP / IP**, http://www.juliobattisti.com.br/artigos/windows/tcpip_p11.asp. Acessado em: 6 de março de 2005.

DI-FCT-UNL Redes de Computadores 2004/2005 (Aulas Práticas), **INTRODUÇÃO À PROGRAMAÇÃO COM SOCKETS**, http://asc.di.fct.unl.pt/rc/aulas-praticas/aulas/aula1/docs/ProgSocketsTCPJava.pdf. Acessado em: 13 de março de 2005.

SANTOS, R.F.T., BARBOSA, R.V., **MINICHAT USANDO SOCKETS EM JAVA**, http://www.portaljava.com. Acessado em: 13 de março de 2005.