Redes de Computadores

Exercício Prático – Programação com Sockets

Prof. Vitor Barbosa Souza
vitorbs@dpi.ufv.br
DPI – Departamento de Informática
UFV – Universidade Federal de Viçosa

Exercício

Crie dois programas com capacidade de se comunicarem entre si através de redes de computadores utilizando soquetes. O primeiro programa deverá funcionar como um servidor de troca de mensagens e arquivos e o segundo programa é o lado cliente.

Cada cliente deverá se comunicar diretamente com o servidor, que deverá se comunicar com diversos clientes de forma paralela com o objetivo de encaminhar as mensagens e arquivos recebidos de um cliente para os demais clientes ativos (com exceção do que enviou) como em um grupo de troca de mensagens.

Especificação

O servidor não tem o objetivo de ser utilizado diretamente por usuários, dessa forma, ele não precisa ler dados escritos pelo usuário no console de execução. O seu objetivo é interpretar as mensagens recebidas e tomar as devidas providências. Em alguns casos, ele pode também mostrar uma mensagem em seu console para fins de log, como um aviso de entrada de novo usuário ou estabelecimento de conexões.

O servidor deve escutar a porta 20000 para troca de mensagens com o cliente utilizando UDP. A troca de arquivos deverá ser realizada por TCP através da porta 20000. O protocolo da aplicação deverá incluir os seguintes tipos mensagens.

Palavra-chave	Descrição
USER:<nome>	Primeira mensagem enviada pelo cliente para informar o <nome> de usuário. O servidor deverá manter uma lista de usuários ativos para que possa encaminhar mensagens recebidas.
ACK	Enviada pelo servidor ao cliente como resposta da mensagem USER. Se ACK não for recebido dentro de 10 segundos, informar que servidor não está disponível.
KEEP	Mensagem “keepalive” enviada pelo servidor a todos os clientes ativos a cada 10 segundos. Ao receber, cada cliente envia uma KEEP ao servidor para informar que está ativo. Se o servidor não receber KEEP de um cliente por duas vezes seguidas, ele retira o cliente da lista de clientes ativos antes de fazer a terceira requisição.
MSG:<texto>	Usada pelo cliente para enviar <texto> para que o servidor encaminhe aos outros clientes.
MSG: <nome>:<texto>	Usada pelo servidor para encaminhar uma mensagem recebida, informando também o nome do cliente que enviou.
LIST	Enviada para ao servidor após o cliente digitar “/list” solicitando uma lista de todos os clientes conectados. A lista é enviada apenas para o cliente que fez a solicitação
FILE<arq>	Com um comando “/file <arq>”, o cliente deve estabelecer uma conexão TCP com o servidor, enviar o arquivo <arq> e finalizar a conexão
INFO: <texto>	Ao receber o arquivo, o servidor deve salvar o mesmo em cache e enviar uma mensagem UDP a todos os clientes informando que o cliente <nome> disponibilizou o arquivo <arq>. A diretiva INFO pode ser utilizada para esta e outras informações.
GET<arq>	No comando “/get <arq>”, o cliente estabelece uma conexão TCP com o servidor, obtém o arquivo <arq> e finaliza a conexão. Exibe erro caso arquivo não esteja disponível. Por simplicidade, o servidor pode manter apenas um arquivo em cache. O envio de outro arquivo sobrescreve o anterior.
BYE	Enviado ao servidor após o comando “/bye” para avisar que o cliente está finalizando. O servidor deve retirar o cliente da lista de clientes ativos ao receber a mensagem. Em seguida, os demais clientes devem ser avisados sobre a saída deste cliente.

Segue um exemplo de várias ações sendo executadas em sequência (os textos em verde representam alguma ação, os textos em vermelho representam uma mensagem trocada entre cliente e servidor, os textos em azul representam um texto digitado pelo usuário e os textos na cor preta representam as mensagens impressas na tela do servidor e de cada cliente):

Servidor	Cliente	Cliente	Cliente
Aguardando conexão	–	–	–
ACK INFO:Ana entrou	Nome de usuário: Ana USER:Ana	–	–
ACK INFO:Bruno entrou	Bruno entrou	Nome de usuário: Bruno USER:Bruno	–
MSG:Ana:olá	olá MSG:olá	Ana disse: olá	–
<responder LIST>		/list LIST Clientes conectados: Ana, Bruno	–
KEEP	KEEP	KEEP	–
ACK INFO:Camila entrou	Camila entrou	Camila entrou	Nome de usuário: Camila USER:Camila
<responder LIST>			/list LIST Clientes conectados: Ana, Beatriz, Camila
<conexão TCP> ***<encerrar conexão>*** INFO:Ana enviou 1.jpg	/file 1.jpg ***<conexão TCP>*** FILE 1.jpg ***<encerrar conexão>***	Ana enviou 1.jpg	Ana enviou 1.jpg
MSG:Ana:tchau	tchau MSG:tchau	Ana disse: tchau	Ana disse: tchau
INFO:Ana saiu	/bye BYE ***<fim do programa>***	Ana saiu	Ana saiu
<conexão TCP> ***<encerrar conexão>***	–	/get 1.jpg ***<conexão TCP>*** GET 1.jpg ***<encerrar conexão>***	
<responder LIST>	–	/list LIST Clientes conectados: Bruno, Camila	
<conexão TCP> ***<encerrar conexão>***	–		/get 1.jpg ***<conexão TCP>*** GET 1.jpg ***<encerrar conexão>***
INFO:Bruno saiu	–	/bye BYE ***<fim do programa>***	Bruno saiu
INFO:Camila saiu	–	–	/bye BYE ***<fim do programa>***

O trabalho pode ser feito individualmente ou em dupla. Para a entrega, os arquivos servidor e cliente devem ser compactados e enviados pelo PVANet por apenas um dos membros da dupla. Colocar o nome dos dois membros da dupla no cabeçalho do código.