Redes de Computadores

Exercício Prático – Programação com Sockets

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

Escreva dois programas em Python, C++ ou Java com capacidade de se comunicarem entre si através de redes de computadores de modo que o primeiro funciona como um servidor de troca de mensagens e arquivos, e o segundo programa é o lado cliente. Você deverá implementar o protocolo de aplicação descrito a seguir utilizando sockets para a troca de mensagens e arquivos.

Especificação do protocolo de aplicação

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.

Como o servidor não tem o objetivo de ser utilizado diretamente por usuários, 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 suportar as seguintes instruções.

Palavra-chave	Descrição
USER:<nome>	Primeira mensagem enviada pela aplicação cliente ao servidor para informar o <nome> de usuário. O servidor deverá manter uma lista de usuários ativos para que possa encaminhar mensagens recebidas.
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 resposta deve ser enviada apenas para o cliente que fez a solicitação
FILE:<arq>\n <dados>	Com um comando “/file <arq>”, o cliente estabelece uma conexão TCP com o servidor, envia o nome do arquivo (<arq>) e o conteúdo do mesmo (<dados>), usando \n para demarcar o fim do nome e o início do conteúdo. Ao final da transferência, a conexão TCP deve ser finalizada.
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 enviadas a um ou mais clientes.
GET:<arq>	No comando “/get <arq>”, o cliente estabelece uma conexão TCP com o servidor, envia o nome do arquivo desejado (<arq>) e, após o recebimento do mesmo, finaliza a conexão. Caso o arquivo não exista no servidor, o próprio servidor pode simplesmente encerrar a conexão.
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. Para facilitar a visualização e compreensão diferentes cores são utilizadas:

- os textos em **verde** descrevem uma ação sendo realizada
- os textos em **vermelho** representam as mensagens do protocolo de aplicação descrito acima, ou seja, são as mensagens trocadas entre cliente e servidor
- os textos em **azul** representam um texto digitado pelo usuário da aplicação
- os textos na cor **preta** representam as mensagens impressas na tela do servidor ou do cliente

Servidor	Cliente	Cliente	Cliente
Aguardando conexão	–	–	–
INFO:Ana entrou	Nome de usuário: Ana USER:Ana	–	–
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	–
INFO:Camila entrou	Camila entrou	Camila entrou	Nome de usuário: Camila USER:Camila
<responder LIST>			/list LIST Clientes conectados: Ana, Bruno, Camila
<aguardando conexão> ***<encerrar conexão>*** INFO:Ana enviou 1.jpg	/file 1.jpg ***<inicia conexão TCP>*** FILE:1.jpg\n ***<envio de fluxo de bits>*** ***<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
<aguardando conexão> ***<envio fluxo de bits>*** ***<encerrar conexão>***	–	/get 1.jpg ***<iniciar conexão TCP>*** GET:1.jpg ***<encerrar conexão>***	
<responder LIST>	–	/list LIST Clientes conectados: Bruno, Camila	
<aguardando conexão> ***<encerrar conexão>***	–		/get foto.jpg ***<iniciar 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 enviados por apenas um dos membros da dupla. Colocar o nome dos dois membros da dupla no cabeçalho do código.