Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computação e Automação
DCA0130 – Redes de Computadores

Trabalho prático 2 – Criação de uma aplicação de bate-papo usando sockets

Entrega: 29/10/2018 23h59

Instruções:

- Trabalho em duplas ou trios;
- A nota deste trabalho será parte da avaliação da Unidade II;
- O trabalho consistirá na implementação de uma aplicação de bate-papo usando sockets;
- A submissão do trabalho deverá ocorrer via SIGAA até a data indicada acima;
- Não será necessária a elaboração de relatório, porém o trabalho deverá ser apresentado pessoalmente ao professor em horário a ser agendado.

Objetivo:

Colocar em prática os conhecimentos adquiridos em sala de aula e praticados em laboratório sobre as camadas de aplicação e de transporte e seus protocolos relacionados. Vocês irão criar uma sala de bate-papo utilizando sockets no modelo cliente/servidor. A implementação deverá ser em linguagem Python. Estão disponíveis scripts de exemplo como base para a implementação de sockets e threads em:

http://www.dca.ufrn.br/~viegas/disciplinas/DCA0130/files/Sockets/

Forma de avaliação:

- Cada dupla irá apresentar e explicar o código fonte/implementação e o funcionamento da aplicação;
- A aplicação desenvolvida será executada em um computador (servidor) e outros computadores/dispositivos (clientes) irão se conectar ao mesmo;
- Inovações e soluções diferenciadas são bem-vindas e terão uma avaliação diferenciada (para melhor);
- Cópias não são admitidas! Evitem usar implementações de outras pessoas, sob a pena de zerar a nota.

Instruções de funcionamento:

Criar uma aplicação cliente e outra servidora, em que:

1. O servidor é “uma sala” de bate-papo em grupo, onde os clientes irão se conectar (não havendo qualquer limite de clientes conectados).
2. Como protocolo de transporte deve ser utilizado o TCP para que haja o controle das conexões.
3. Os clientes conectados poderão enviar e receber mensagens para o servidor (ou seja, para todos “da sala”).

4. Quando um cliente envia mensagens para “a sala”, o servidor deverá encaminhá-las para todos os demais clientes conectados, bem como mostrá-las em sua tela (terminal). Neste ponto, tanto os clientes quanto o servidor devem exibir as mensagens nas suas respectivas telas.
5. Quando um cliente se conecta ao servidor, este solicitará um apelido (*nickname*), que será utilizado para identificar o cliente na comunicação:
    a. Quando um cliente se conecta ao servidor, deverá ser apresentada uma mensagem de que o mesmo entrou na sala: `nick-do-cliente entrou...`
    b. Da mesma forma, quando um cliente sair da sala: `nick-do-cliente saiu!`
    c. Quando um cliente enviar uma mensagem, deverá ser apresentado em tela/terminal da seguinte forma: `nick-do-cliente escreveu: mensagem`
    d. Um cliente poderá alterar o seu nome a qualquer momento e o servidor deverá comunicar essa alteração: `nick-do-cliente agora é novo-nick-do-cliente`
6. O servidor deverá exibir uma lista com os clientes a ele conectados, mostrando ⟨NOME, IP, PORTA⟩. Esta informação DEVE ser solicitada tanto na tela do cliente, quanto na tela do servidor.
7. Um cliente, além de se conectar ao servidor para um bate-papo em grupo, também deve ser capaz de se conectar a outro cliente por meio de conexão privada (“uma sala privada”):
    a. Esta conexão privada deve ter como intermediário o servidor, ou seja, o tráfego do cliente ‘A’ é enviado para o servidor e este encaminha para o cliente ‘B’ (e vice-versa);
    b. Se um cliente estiver conectado ao servidor de bate-papo em grupo e iniciar uma conexão privada com um outro cliente, as mensagens trocadas entre esses clientes não devem ser exibidas na tela do servidor. Neste caso, quando um cliente ‘A’ desejar se comunicar de forma privada com um cliente ‘B’, ‘A’ solicitará a ‘B’ e este receberá o pedido de solicitação, aceitando-o ou rejeitando-o. Caso o pedido seja aceito, ‘A’ e ‘B’ iniciarão uma sala privada. Caso o pedido seja rejeitado, uma mensagem de rejeição deverá ser enviada para o cliente que solicitou;
    c. Quando uma conexão privada for encerrada, ambos os clientes deverão retornar para a sala de bate-papo em grupo. Neste caso, tanto ao iniciar o privado quanto ao voltar para o grupo, os clientes utilizarão o mesmo nome (*nickname*) previamente digitado;
    d. Quando dois clientes estão em um bate-papo privado, a lista de usuários conectados deve exibir os nomes dos mesmos, porém com o indicador de que estão em privado. Por exemplo:

        Lista:

        Carlos
        Professor (privado)
        Aluno (privado)
8. Para que o servidor seja capaz de receber e responder às conexões de múltiplos clientes, o mesmo deverá ser implementado utilizando **threads**. Neste caso,

para cada cliente que se conectar ao servidor deverá ser criada uma ou mais threads para gerenciar o envio e/ou o recebimento das mensagens.

9. A comunicação entre cliente e servidor deverá ser regida por um protocolo de aplicação, a ser desenvolvido. Cada mensagem trocada entre o servidor e o cliente deve conter um cabeçalho com os campos abaixo. As mensagens devem ser criadas neste formato, enviadas por meio do socket e interpretadas por quem as recebe.

1 octeto	4 octetos	4 octetos	6 octetos	1 octeto	até 40 octetos
Tamanho da mensagem	Endereço IP de origem	Endereço IP de destino	Nickname do usuário	Comando	Dados

   a. Tamanho da mensagem: deverá conter o tamanho (em bytes) da mensagem a ser enviada, incluindo o cabeçalho;
   b. Endereços IP de origem e destino: deverá conter os endereços IP das partes comunicantes;
   c. Nickname do usuário: deverá indicar o nickname do usuário de destino. Para os casos em que o destino é “para todos” ou “para o servidor” fica a seu critério a escolha do que deverá conter este campo;
   d. Comando: indica qual é o comando enviado (ex: `lista()`, `nome()`, `sair()`, etc.).
   e. Dados: conteúdo das mensagens.

Instruções específicas:

1. O **servidor** deve:
   a. Aceitar conexões dos clientes (sem limites);
   b. Criar uma ou mais threads para cada cliente conectado;
   c. Solicitar um nome (*nickname*) a cada cliente que se conectar;
   d. Permitir que todos os clientes conectados enviem e recebam mensagens;
   e. Permitir que os clientes possam iniciar comunicações privadas por meio do comando `privado(*)`, onde `*` será o nome do cliente;
   f. Exibir a lista de clientes conectados por meio do comando `lista()`, mostrando o ⟨NOME, IP, PORTA⟩ de cada um;
   g. Aceitar alterações de nomes dos clientes por meio do comando `nome(*)`, onde `*` será o novo nome desejado;
   h. Encerrar todos os clientes quando o servidor for encerrado utilizando o comando `sair()`.
2. O **cliente** deve:
   a. Conectar-se ao servidor;
   b. Escolher um nome (*nickname*) para se identificar no bate-papo por meio do comando nome(`*`), onde `*` é o nome desejado;
   c. Enviar e receber mensagens em grupo;
   d. Requisitar uma comunicação privada por meio do comando `privado(*)`, onde `*` será o nome do cliente a se conectar;
   e. Requisitar a lista de clientes por meio do comando `lista()`;

f. Alterar a qualquer momento o seu nome por meio do comando `nome(*)`, onde `*` será o novo nome desejado;
g. Encerrar a sua aplicação ao digitar `sair()`.

Pesos na avaliação das tarefas:

- Os itens 1 a 6 correspondem a 20% da avaliação do projeto.
- Os itens 7 e 8 correspondem a 30% da avaliação do projeto.
- O item 9 corresponde a 30% da avaliação do projeto.
- O correto funcionamento e as inovações no projeto complementarão os 20% restantes. É importante lembrar que a não implementação de algum item compromete o peso atribuído a este último.