Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computação e Automação
DCA0113 – Redes de Computadores

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

Entrega: 31/05/2017 23h59

Instruções:

- Trabalho em dupla (sendo o mesmo grupo formado anteriormente);
- A nota deste trabalho será 40% 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 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 na linguagem Python, por uma razão de simplicidade. Estão disponíveis scripts de exemplo como base para a implementação de sockets e threads em:

http://www.dca.ufrn.br/~viegas/disciplinas/DCA0113/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 serão bem-vindas e terão uma avaliação diferenciada (para melhor);
- Cópias não são admitidas! Evitem usar implementações de outras pessoas.

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. Deve ser usado o protocolo de transporte TCP para que haja o controle efetivo da conexão.
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).

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, 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 pode ser solicitada tanto pelo cliente, quanto na tela do servidor.
7. O servidor deverá implementar uma função *anti-flood*, na qual um cliente não deve ser capaz de enviar mais de 5 mensagens em um intervalo de 2 segundos, ou seja, ao digitar 5 mensagens seguidas dentro de 2 segundos, o cliente ficará inibido de enviar novas mensagens durante 10 segundos. Deverá ser emitido um alerta, apenas na tela do cliente, de que o limite de mensagens foi excedido e que deverá aguardar 10 segundos.
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.

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. Fazer controle *anti-flood* de 5 mensagens em um intervalo de 2 segundos;
    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;
    c. Enviar e receber mensagens;
    d. Requisitar a lista de clientes por meio do comando `lista()`;
    e. Alterar a qualquer momento o seu nome por meio do comando `nome(*)`, onde * será o novo nome desejado;
    f. Encerrar a sua aplicação ao digitar `sair()`.