Trabalho Prático 1 de Redes de Computadores:

Enquadramento de Dados por DCCNET

Breno Rodrigues Marques da Silva, Davi Faria de Assis Beltrão

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

Av. Antônio Carlos, 6627 - Prédio do ICEx, Pampulha, Belo Horizonte, Minas Gerais, Brasil.
CEP: 31270-901

{brenorms,davibeltrao}@dcc.ufmg.br

1. Introdução

Neste trabalho desenvolvemos uma aplicação que realiza troca de mensagens através do Python 3. Ela possui dois componentes: um o cliente e o outro o servidor. Os usuários devem utilizar o módulo de cliente para ser conectar a um servidor central. A comunicação será feita através de um protocolo feito sobre comunicação TCP.

Nesse protocolo, toda mensagem possui um cabeçalho, com quatro campos:

- Tipo de mensagem (2 bytes): o tipo de mensagem enviada.
- Identificador de origem (2 bytes): identificador do cliente de origem.
- Identificador de destino (2 bytes): identificador do cliente de destino.
- Número de sequência (2 bytes): identificador de cada mensagem individual. O número de sequência é incrementado toda vez que o cliente envia uma nova mensagem. Se o programa atingir o maior número possível (65535), ele deve recomeçar do zero.

A seguir estão listados os tipos de mensagens do protocolo e seus identificadores:

- OK (1): todas as mensagens devem ser respondidas com uma mensagem de OK ou de ERRO. Essas mensagens carregam o mesmo número de sequência da mensagem que está sendo confirmada.
- ERRO (2): enviada como confirmação, ao invés do OK em situações onde a mensagem não pode ser aceita.
- OI (3): primeira mensagem que o cliente envia ao servidor. Ela é usada para que o cliente se identifique ao servidor.
- FLW (4): mensagem do cliente ao servidor para indicar que ele vai se desconectar.
- MSG (5): mensagem de texto originada em um cliente e enviada para outro cliente. Caso o destino seja 0, a mensagem é enviada para todos os clientes conectados ao servidor.
- CREQ (6): mensagem do cliente para o servidor para que ele retorne uma mensagem CLIST.
- CLIST (7): mensagem de retorno do CREQ, contendo uma lista de clientes conectados.

2. Implementação

O principal desafio de implementação foi o uso da função *select()* para selecionar quando devem ser feitas operações de leitura e escrita em quais *file descriptors*.

O cliente realiza o *select* sobre o *file descriptor* do socket do servidor e o *stdin*. Cada *file descriptor* só é selecionado quando há alguma entrada, logo o *stdin* só é selecionado quando o usuário digita uma entrada, e o socket só é selecionado quando há algo a ser recebido. O socket só é selecionado para escrita quando há algo a ser enviado por ele.

O servidor coloca as mensagens a serem enviadas em uma fila, e elas são enviadas uma a uma a medida que os seus respecitos sockets são selecionados para escrita.

Na implementação as mensagens de OK são esperadas imediatamente após os envios. A única exceção é a mensagem de tipo 5 (MSG), que possui um *timeout*, como descrito na

especificação do trabalho. Caso essa exceção ocorra, o cliente que falhou em confirmar é retirado do servidor.

Outra exceção é a mensagem do tipo 6 (CREQ), que não possui confirmação por OK ou ERRO, e só por CLIST.

3. Execução

O programa do Cliente se encontra no arquivo *cliente.py* e o do Servidor no arquivo *server.py*.

O cliente deve ser executado com dois argumentos: o endereço de IP do servidor ao qual se deseja conectar e a porta que será utilizada. O servidor possui um argumento: a porta que deve ser utilizada para receber conexões.

Será impresso na saída padrão a forma como as mensagens devem ser escritas. Primeiro um inteiro com o identificador da mensagem a ser enviada (4, 5 ou 6), seguido do destino e da mensagem (redundantes a não ser que a mensagem seja do tipo 5).

4. Conclusão

Neste trabalho implementamos uma aplicação para troca de mensagens sobre TCP. Os clientes conectam a um servidor central e as várias mensagens são repassadas através do servidor.