Segundo Exercício de Programação: RPC

Giovanna Louzi Bellonia[1]

[1] Matrícula: 2017086015

Resumo. *Relatório sobre as decisões de projeto do trabalho da matéria de Fundamentos de Sistemas Paralelos e Distribuídos.*

1. Interface

A interface do gRPC foi definido no arquivo services.proto e chamado de DoStuff. Ele possui duas funções RPC: get_service_port e get_service_description.

Tanto a get_service_port quanto a get_service_description recebem como parâmetro um tipo ServiceName, que tem um único elemento 'name' do tipo String. A primeira função RPC retorna um tipo ServicePort que tem um único elemento 'port' de tipo int32. Já a segunda função possui três elementos: 'protocol', 'aliases' e 'comments', todos do tipo string.

2. Servidor

O servidor foi definido no arquivo server.py e utiliza o pool de threads para funcionar. As chamadas de rpc são passadas para o servidor com a ajuda da classe DoStuff presente no mesmo arquivo. Para inicializar essa classe, é necessário passar como parâmetro um arquivo txt que será utilizado para criar um dicionário services_dict que auxiliará na busca dos dados de um determinado serviço a partir do seu nome. Cada serviço salvo nesse dicionário possui um outro dicionário com as chaves 'port', 'protocol', 'aliases' e 'comments', podendo as duas últimas terem seu valor salvo como strings vazias se esses dados não existirem no arquivo txt.

Além disso, a classe DoStuff possui duas funções que serão chamadas pelos clientes: get_service_port e get_service_description. Ambas recebem como parâmetro o nome do serviço e utilizam do dicionário definido na inicialização da classe para fazerem suas buscas. A primeira função retorna o valor da porta desse serviço ou -1 caso ele não exista. Já a segunda retorna os valores de protocol, aliases e comments ou, caso o serviço não exista, string vazia para esses valores. Essas duas funções printam no servidor uma linha que passa as informações da função, sendo elas o endereço do cliente (client address), o nome da função chamada (command called), o nome do serviço passado como parâmetro (service name) e os resultados encontrados que serão passados para o cliente.

3. Cliente

O cliente foi definido no arquivo client.py e chama as duas funções do servidor de acordo com o parâmetro recebido na chamada do MakeFile. Primeiramente ele chama a função get_service_port e printa o resultado encontrado e depois chama a função get_service_description e também printa o resultado recebido, colocando antes de cada tipo o nome dele para maior clareza (por exemplo, coloca "protocol = "antes de printar o valor de protocol recebido).

4. Makefile

O Makefile permite a chamada do make run_server passando os parâmetros arg1 e arg2 para serem utilizados no servidor para definir sua porta e passar o endereço do txt para ser utilizado na classe DoStuff, respectivamente.

Além disso permite a chamada do make run_client passando os parâmetros arg1 e arg2 para serem utilizados no cliente, respectivamente, para definir o endereço do cliente e o nome do serviço que vai ser passado para as funções chamadas dentro do cliente.

Antes de cada chamada, o MakeFile faz um comando que garante que os arquivos necessários para o funcionamento do grpc estão definidos.

5. Bibliografia

O código feito foi baseado no exemplo dado pelo professor da disciplina em uma das vídeo aulas. Além disso foi utilizado os conhecimentos adquiridos nessa e em outras matérias, além do link disponibilizado na descrição do trabalho para entendimento de como os serviços são representados.