Aplicação de Sockets em Python 3 para Estabelecimento de Conexão para Troca de Mensagens

Clara Bettencourt[1], Gabriel Antunes[1], Gabriela Amaral[1], João Pedro Macabu[1], Mariana Moraes[1]

[1]Coordenação de Engenharia de Controle e Automação – Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro – RJ – Brasil

clarabette@poli.ufrj.br, g.queirozantunes@poli.ufrj.br,
gabii.a.alcantara@poli.ufrj.br, jpcosta97@poli.ufrj.br,
marianamoraesfs@poli.ufrj.br

Abstract. *In this work, a protocol was made for use in message exchange. A client-server model was developed, allowing clients to use a common server as a bridge for communication, while it stores the conversation log as a precautionary measure.*

Resumo. *Neste trabalho, foi feito um protocolo para uso em troca de mensagens. Um modelo cliente-servidor foi desenvolvido, permitindo que clientes usem um servidor comum como ponte para a comunicação, enquanto este último armazena o registro da conversa como uma medida de precaução.*

1. Modelo Cliente-Servidor

Neste tópico, serão abordados os códigos *cliente.py* e *servidor.py*, os métodos usados para a construção de ambos, e suas fontes.

Acerca da estrutura, este modelo é composto de dois componentes, o cliente (que executa a ação de conectar-se ao servidor, propiciando uma interface com o usuário, para o envio e o recebimento de mensagens) e o servidor (que se encarrega de estabelecer a conexão entre clientes em uma mesma rede). Estes dois componentes podem ser observados, respectivamente, nos códigos *cliente.py* e *servidor.py*.

1.1. Cliente

Como ponto de partida, tomemos o código *cliente.py*, onde será estabelecida uma conexão entre o usuário executor do programa e o servidor.

1.1.1. Código cliente.py

Primeiramente, foram definidas as variáveis globais tam_cabe, host, port, end e nome.

- tam_cabe é o tamanho do cabeçalho a ser utilizado (pré-definido)
- host é o número de IP do servidor
- port é o número da porta a ser utilizada

- end consiste no endereço gerado pela junção (host, port)
- nome é uma variável que recebe o nome de usuário desejado do usuário executor do código

Definiu-se o cliente como uma instância do objeto *socket* proveniente do módulo *socket* da linguagem Python, com conexão a ser estabelecida via protocolos IPv4 (AF_NET) e TCP (SOCK_STREAM). Esta instância foi armazenada em uma variável denominada conn. Através do método connect(end) do objeto conn, é então estabelecida a conexão ao servidor cujo endereço foi fornecido por end.

A partir daí, o programa codifica o nome de usuário desejado e o tamanho do cabeçalho, e os envia ao servidor para que o cliente possa se juntar a sessão de conversa aberta pelo servidor.

Através de um *while loop* infinito, o programa continuamente pede por *input* do usuário na forma das mensagens a serem transmitidas para os outros participantes da conversa. Estes *inputs* são similarmente codificados e enviados ao servidor, que então os retransmite aos demais clientes a ele conectados.

Um segundo *while loop* infinito atua como receptor das mensagens provenientes do servidor, decodificando-as para que sejam exibidas de maneira legível ao usuário. Caso não haja recebimento de mensagens neste *loop*, o programa assume que a conexão ao servidor foi encerrada e, após notificar o usuário deste ocorrido, o programa se encerra.

Foram também acrescentadas, ao fim do código, medidas para lidar com erros em conexões sem bloqueio. Em caso de um erro de leitura devido a *input / output*, o programa alerta o usuário deste ocorrido e procede sua execução. Para demais erros, o programa alerta o usuário deste ocorrido e, em seguida, se encerra.

1.2. Servidor

Para o código do cliente funcionar, precisamos de um meio por onde as mensagens serão transmitidas, logo, surge a necessidade de um servidor.

1.2.1. Código servidor.py

Começamos o código abrindo um documento de texto de nome ‘message_log’ com a configuração de leitura e escrita. Neste arquivo serão salvas as mensagens enviadas por este servidor. Temos, logo após, as mesmas definições usadas no código do cliente (tam_cabe, host, port e end) e a criação da instância de socket que receberá as conexões (serv).

Usamos então o método *setsockopt* para zerar o *TIME_WAIT*, de forma a prevenir que caso o usuário utilize o atalho Ctrl+C para interromper o programa, o programa seja fechado sem que a porta associada ao servidor seja também fechada.

Definimos então a função *receive_message* que retorna False caso o tamanho de cabeçalho da mensagem seja diferente daquele definido previamente. Caso contrário, ela define o tamanho da mensagem.

Criamos um *while loop* infinito para receber as mensagens, que começa lendo os sockets e separando algumas exceções, colocando o resto em uma lista vazia. Para cada socket, é realizado um conjunto de ações.

Caso um novo usuário se conecte ao servidor, ele é aceito e adicionado à lista de sockets, e seu nome é enviado aos demais clientes alertando-os de sua entrada.

Caso o usuário já exista na lista, ele recebe a mensagem enviada pelo mesmo, e se ele se desconectar, os demais clientes são similarmente alertados de que sua conexão foi encerrada.

No momento em que todos os usuários tiverem se desconectado, o servidor salva todas as mensagens enviadas no arquivo de texto e o programa encerra sua execução.

2. Referências

Após formulada a base do protocolo, foram utilizados alguns códigos prontos encontrados na internet para auxiliar em sua implementação. Os endereços das páginas fontes são fornecidos abaixo.

Estes códigos serviram como base para o polimento das mensagens e a apresentação do aplicativo, melhorando também a organização e o entendimento sobre como proceder para executar o conjunto cliente-servidor com sucesso.

2.1 Sites

https://pythonprogramming.net/server-chatroom-sockets-tutorial-python-3/?completed=/pickle-objects-sockets-tutorial-python-3/

https://pythonprogramming.net/client-chatroom-sockets-tutorial-python-3/?completed=/server-chatroom-sockets-tutorial-python-3/

https://www.weblink.com.br/blog/tecnologia/conheca-os-principais-protocolos-de-internet/

https://wiki.python.org.br/SocketBasico

https://blog.4linux.com.br/socket-em-python/

https://docs.python.org/3/howto/sockets.html

https://realpython.com/python-sockets/

2.2 Vídeos

https://www.youtube.com/watch?v=6sHGBXwkFQU&feature=youtu.be

https://www.youtube.com/watch?v=Rvfs6Xx3Kww

https://www.youtube.com/watch?v=Lbfe3-v7yE0