Ciências ULisboa

Aplicações Distribuídas
2021/22

Guião de apoio 1
Introdução aos *sockets* TCP e aos programas cliente-servidor

---

1. Introdução ao tema

Nesta aula vamos realizar alguns exercícios em *Python* relacionados com *sockets*, base fundamental sob a qual os sistemas e aplicações distribuídas são construídos.

Conforme visto na aula TP, o uso de *sockets* para interligação de dois processos requer a chamada de uma série de funções nos processos intervenientes. A Figura 1 resume esse fluxo (ver slides da aula TP 1 para mais detalhes sobre as funções invocadas).

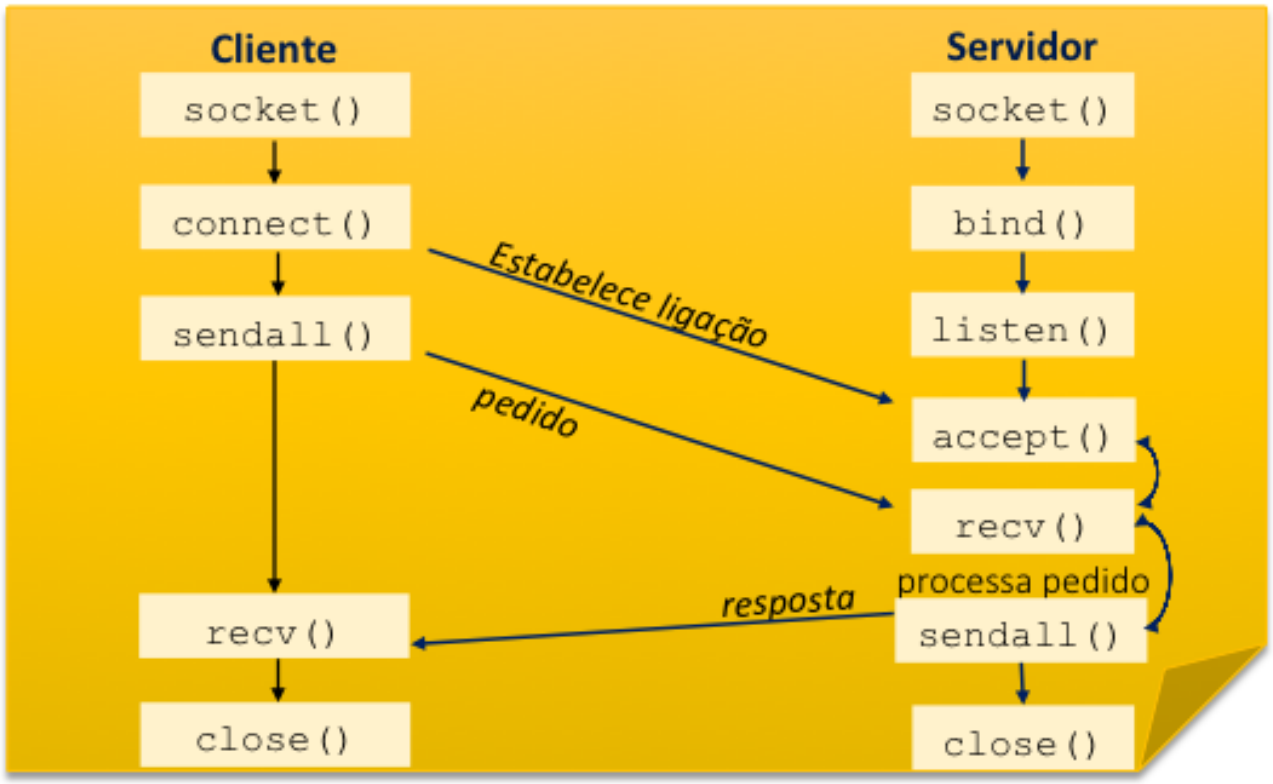

Figura 1. Cliente/servidor com ligação TCP.

O exemplo a seguir apresenta um cliente que envia uma mensagem a um servidor que apenas imprime essa mensagem e responde ao cliente.

Cliente (*cliente.py*)

```
import socket as s

HOST = '127.0.0.1'
PORT = 9999

sock = s.socket(s.AF_INET, s.SOCK_STREAM)

sock.connect((HOST, PORT))

sock.sendall(b'Vamos aprender isto!')
resposta = sock.recv(1024)

print('Recebi %s' % resposta)

sock.close()
```

Servidor (*servidor.py*)

```
import socket as s

HOST = '' #pode ser vazio, localhost ou 127.0.0.1
PORT = 9999

sock = s.socket(s.AF_INET, s.SOCK_STREAM)
#sock.setsockopt(s.SOL_SOCKET, s.SO_REUSEADDR, 1)
sock.bind((HOST, PORT))
sock.listen(1)
(conn_sock, (addr, port)) = sock.accept()

print('ligado a %s no porto %s' % (addr,port))

msg = conn_sock.recv(1024)
print('recebi %s' % msg)
conn_sock.sendall(b'Aqui vai a resposta')

sock.close()
```

Entre muitos aspetos importantes omitidos para tornar o exemplo simples (alguns dos quais abordaremos a seguir), é de salientar que este código não trata erros.

Num sistema distribuído, os erros podem acontecer a qualquer momento de uma interação, já que falhas parciais podem afetar apenas alguns processos do sistema. Assim, é importante que todas as interações entre processos estejam sempre dentro de blocos *try/except*.

---

2. Exercícios fundamentais

1. Copie os programas cliente e servidor apresentados neste guião e execute-os no computador do laboratório. Qual o output do cliente e do servidor?

2. Modifique os programas para que os endereços e portos utilizados sejam passados por linha de comando tanto no cliente quanto no servidor. Além disso, faça com que a mensagem enviada para o servidor seja lida como input do utilizador.

3. Crie um módulo de nome ***sock_utils.py*** onde irá implementar algumas funções que poderão vir a ser reutilizadas em vários programas. Inicialmente pretendem-se as seguintes funções:

- **`listener_socket = create_tcp_server_socket(address, port, queue_size)`**
  Esta função serve para criar uma socket de servidor TCP onde poderão posteriormente ser aceites ligações de clientes.
  address será o endereço anfitrião à qual a socket ficará vinculada.
  port será a porta onde o servidor atenderá novos pedidos de ligação.
  queue_size define o número máximo de pedidos de ligação em espera. (ver função listen dos objetos da classe socket).
  listener_socket será a socket de servidor TCP criada.

- **`client_socket = create_tcp_client_socket(address, port)`**
  Esta função serve para criar uma socket de ligação para o cliente comunicar com um servidor.
  address será o endereço do servidor onde o cliente se ligará.
  port será a porta onde o servidor atende pedidos de ligação.
  client_socket será a socket de ligação que o cliente usará para comunicar com o servidor.

- **`dados_recebidos = receive_all(socket, length)`**
  Esta função deverá receber no máximo lenght bytes através da socket. Devem considerar o caso de a socket remota fechar a ligação antes de enviar o número esperado de bytes.
  socket será a socket de ligação para ler os dados.
  lenght determina o número de bytes que devem ser lidos e devolvidos.

Altere os programas do Exercício 2 de forma a importar e utilizar estas funções sempre que for adequado.

4. Modifique o programa anterior para que a mensagem enviada pelo cliente seja inserida num dicionário no servidor. A chave deverá ser um número inteiro inicializado a zero, que deverá ser incrementado a cada nova inserção. Faça outra alteração para que o servidor nunca termine a execução e volte a esperar uma nova ligação logo que a ligação existente com um cliente termine.
(Ver https://docs.python.org/3/tutorial/datastructures.html#dictionaries)

5. Modifique o programa anterior para que ele altere o seu comportamento de acordo com a *string* recebida:

- Caso receba a *string* "GET <num>", onde <num> é um número inteiro, retorna a *string* associada à chave <num> no dicionário do servidor, ou a *string* "chave inexistente" caso esse número não seja chave no dicionário.
- Caso receba a *string* "LIST", retorna uma grande *string* com todas as *strings* existentes no dicionário concatenadas e separadas por vírgulas. Se o dicionário estiver vazio, deve retornar a *string* "dicionário vazio".
- Outas *strings* deverão ser inseridas no dicionário e a resposta deve ser uma *string* com a chave criada.

Pergunta: em que medida a concretização dessas operações limita o que podemos guardar na lista?

6. Modifique o programa anterior para evitar que o cliente tenha de ser reexecutado, incluindo a conexão com o servidor, sempre que queira manipular o dicionário no servidor. Isto pode ser feito, por exemplo, colocando as operações de leitura do input, envio da mensagem e recepção da resposta dentro de um ciclo. Faça com que o programa só saia do ciclo se o cliente introduzir a *string* "EXIT" (que não deve ser enviada ao servidor).

7. Execute vários programas clientes concorrentemente para manipular uma lista num único servidor. O que observou? Com o que aprendemos até aqui, modifique o programa para que eles efetivamente manipulem a lista em paralelo.

3. Exercícios complementares

Os alunos que terminem com sucesso os exercícios fundamentais são convidados agora a resolver um exercício adicional:
1. Modifique os programas desenvolvidos para inserir uma *string* s no dicionário através da mensagem "ADD,*s*". Adicionalmente, permita que se removam elementos da lista através da mensagem "REMOVE,<num>". Note que com essas modificações o programa passa a não inserir automaticamente qualquer s*tring* recebida, respondendo apenas às mensagens "ADD,*", "REMOVE,*", "GET" e "LIST".

4. Bibliografia e outro material de apoio

https://docs.python.org/3/library/socket.html
https://docs.python.org/3/tutorial/datastructures.html#dictionaries