CSC – Ciência da Computação

Introdução a API em sockets – Formativa 2

PARTE 1 – Definições e Exemplos

Socket pode ser vista como uma API (Application Programming Interface) para desenvolvimento de software que se comunica em rede.

Existem 3 tipos de sockets:

- TCP (ou STREAM)
- UDP (ou DATAGRAM)
- IP (ou modo RAW - sem protocolo de transporte)

1) Exemplo de como fazer a criação do SOCKET

```
#!/usr/bin/env python3

import socket
 …
 // socket.socket(int familia, int tipo);
  socket.socket (socket.AF_INET, socket.SOCK_STREAM);
  // OU
  socket.socket (socket.AF_INET, socket.SOCK_DGRAM);
  // OU
  socket.socket (socket.AF_INET, socket.SOCK_RAW);
```

As opções para família são:
AF_INET (PF_INET) = IPv4
AF_INET6 (PF_INET6) = IPv6

As opções para tipo são:
SOCK_STREAM = TCP
SOCK_DGRAM = UDP
SOCK_RAW = RAW (encapsulamento IP)

2) EXEMPLO DE BIND (**ServidorTCP.py**)

```
#!/usr/bin/env python3

import socket
import sys

HOST = '127.0.0.1'  # localhost = esta máquina
PORT = 9999         # portas abaixo de 1023 exigem permissão de root

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
   s.bind((HOST, PORT))
except:
  print('# erro de bind')
  sys.exit()

s.listen(5)

print('aguardando conexoes em ', PORT)
conn, addr = s.accept()

print('recebi uma conexao de ', addr)

#-------------------------
# insira aqui o código para tratar uma conexao
#-------------------------

print('o cliente encerrou')
conn.close()
```

A sequência típica de um servidor é formada por três chamadas:

a) bind: solicita uma porta para o S.O.
b) listen(N): autoriza o S.O. a receber até N conexões pendentes
c) accept: solicita ao S.O. uma conexão para ser tratada pela aplicação

A função accept retorna dois valores:

conn: representa a conexão com o cliente
addr: traz os endereços do cliente (IP e Porta)

Um HOST vazio, isto é HOST='' vincula o socket a todos os endereços IP do host, envitando que o programa fique dependente de um IP específico.

3) EXEMPLO DE CONNECT (**ClienteTCP.py**)

```
#!/usr/bin/env python3

import socket
import sys

HOST = '127.0.0.1'  # IP do servidor
PORT = 9999         # Porta do servidor

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
   s.connect((HOST, PORT))
except:
  print('# erro de conexao')
  sys.exit()

#--------------------------
# insira aqui o codigo para tratar uma conexao
#--------------------------
```

Esta é a sequência típica de um "connect" de cliente TCP.

A porta do servidor é um número inteiro.

O ip do servidor é um string em notação decimal pontuada.

O servidor precisa ser iniciado antes do cliente, ou a opção connect retornará erro.

4) EXEMPLO TRANSMISSÃO E RECEPÇÃO TCP:

```
// s: objeto socket criado anteriormente no exemplo 3

// TRANSMISSÃO

print('digite o texto a ser transmitido:')

while True:
   try:
      line = input()
      if not line:
         print('linha vazia encerra o programa')
         break
   except:
         print('programa abortado com CTRL+C')
         break

   data = bytes(line, 'utf-8') #converte string para bytes
   tam = s.send(data)

   print('enviei ', tam, 'bytes')

   print(data)     // s: objeto socket criado anteriormente


// RECEPÇÃO

while True:
  data = conn.recv(1024)
  print('recebi ', len(data), ' bytes')

  if not data:
    break
  print(data)

print('a conexao foi encerrada')
conn.close()
```

PARTE 2 – Exercícios

1) EXERCíCIO 1:

Crie o programa (**ServidorTCP.py**)) usando o exemplo 2

Inclua um while True para que o programa receba várias conexões sem encerrar.

DICA: O while no Python é indentado

```
while True:
   conn, addr = s.accept()
   print('recebi uma conexao de ', addr)
```

Com o servidor ativo, execute uma conexão utilizando o Putty no modo **RAW**.

Anote o valor de porta mostrado pelo programa servidor.

Encerre a conexão o Putty, e repita os testes para obter mais dois valores de porta.

Responda:

a) Indique os valores das portas obtidas com o teste (use o utilitário netstat)
b) Captura o tráfego com wireshark e explique o encapsulamento de um frame

2) EXERCÍCIO 2:

Altere o programa servidor para que a porta TCP seja passada seja passada pelo usuário:

DICA: a porta precisa ser convertida de string para inteiro

```
print('Entre com a porta do servidor')
porta = int(input())
```

Execute dois servidores a partir do console do Windows:

```
cmd
C:\>start python3 servidorTCP2.py
C:\>python3 servidorTCP2.py
```

Responda:

a) Indique o que acontece quando você tenta abrir dois servidores simultaneamente na mesma porta

3) EXERCÍCIO 3:

Altere o programa servidor.py para que ele imprima as mensagens recebidas do cliente.

DICA: utilize o código de RECEPCAO fornecido no exemplo 4.

**** Observe que o código deve estar indentado dentro do while ****

Responda:

a) Indique os valores das portas obtidas com o teste (use o utilitário netstat)
b) Captura o tráfego com wireshark e explique o encapsulamento de um frame
c) Digite alguns strings no putty e veja o que aparece na interface do servidor
d) Analise o payload das mensagens via wireshark

4) EXERCÍCIO 4:

Crie o programa clienteTCP.py usando o exemplo fornecido no exemplo 3.

Complete o código usando a seção TRANSMISSÃO fornecida no exemplo 4.

Responda:

a) Indique os valores das portas obtidas com o teste (use o utilitário netstat)
b) Captura o tráfego com wireshark e explique o encapsulamento de um frame
c) Digite alguns strings no cliente e veja o que aparece na interface do servidor
d) Analise o payload das mensagens via wireshark