# Relatório de Redes de Computadores

*Protocolo HTTP*

Aluno: Lukas Maximo Grilo Abreu Jardim

Docente: Carlos Viegas

Ciência e Tecnologia/Engenharia da Computação UFRN

Nesse trabalho, foi simulando uma comunicação entre client e servidor através do uso de sockets – fluxo de comunicação entre aplicativos de rede - , utilizando a linguagem Python. Para isso foi utilizado o seguinte código:

```
import socket

import os as o

import string as st

# definicao do host e da porta do servidor

HOST = '' # ip do servidor (em branco)

PORT = 8082 # porta do servidor

comando = b'GET'

file = b'/index.html'

# cria o socket com IPv4 (AF_INET) usando TCP (SOCK_STREAM)

listen_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


# mantem o socket ativo mesmo apos a conexao ser encerrada (faz o
reuso do endereco do servidor)

listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)


# vincula o socket com a porta (faz o "bind" do servidor com a porta)

listen_socket.bind((HOST, PORT))


# "escuta" pedidos na porta do socket do servidor

listen_socket.listen(1)


# imprime que o servidor esta pronto para receber conexoes

print ("Serving HTTP on port %s ..." % PORT)

#print (comando ==b'GET')

while True:

    # aguarda por novas conexoes

    client_connection, client_address = listen_socket.accept()

    # o metodo .recv recebe os dados enviados por um cliente atraves
do socket
```

```
    request = client_connection.recv(1024)
    # imprime na tela o que o cliente enviou ao servidor
    #print ("req: %s" % request.split()[1])
    #print (str(o.path.exists("./index.html")))
    # declaracao da resposta do servidor

    arquivo = request.split()[1]
    print(arquivo)
    filet = (str(arquivo).split("b'")[1])
    file1 = filet.split('l')[0]
    print(file1)
    print(arquivo == b'/')
    print(o.path.isfile(file1))

    if (request.split()[0] == comando): (1)
        if(arquivo == str(b'/')):
            if (o.path.exists(file) is True): (2)
                http_response = b"""\
                Content-Type: text/html\n
                %s/HTTP/1.1 200 OK

                 """ % comando

                # servidor retorna o que foi solicitado pelo cliente
(neste caso a resposta e generica)
                client_connection.send(http_response)
                client_connection.send(open('index.html'), 'w')

                # encerra a conexao

        else:
```

```
            if (o.path.exists(file1) is True): (3)
                http_response = b"""\
                %s/HTTP/1.1 200 OK

                pagina especificada
                 """ % (request.split()[0] + arquivo)
                # servidor retorna o que foi solicitado pelo cliente
(neste caso a resposta e generica)
                client_connection.send(http_response)
                client_connection.send(open(file1), 'w')

                # encerra a conexao

            else: (4)
                #c, (client_host, client_port) = socket.accept()
                http_response = b"""\

                HTTP/1.1 404 Not Found \r\n\r\n
                <html>
                    <head>
                        <title>
                            Page Not Found
                        </title>
                        <meta charset='utf-8'>
                    </head>
                    <body>
                        <h1>
                            404 Not Found
                        </h1>
                    </body>
                </html>
```

```
                \r\n
                """


                 # servidor retorna o que foi solicitado pelo cliente
(neste caso a resposta e generica)
                client_connection.send(http_response, 'w')
                #arq_html.write(http_response)
                # encerra a conexao


    else:
        #c, (client_host, client_port) = socket.accept()


        http_response = b"""\
        HTTP/1.1 400 Bad Request \r\n\r\n
        <html>
            <head>
                <title>
                    Error
                </title>
                <meta charset='utf-8'>
            </head>
            <body>
                <h1>
                    400 Bad Request
                </h1>
                <h2>
                    Page not found
                </h2>
            </body>
        </html>
        \r\n
```

```
        """


        # servidor retorna o que foi solicitado pelo cliente (neste
caso a resposta e generica)

        client_connection.send(http_response, 'w')

        # encerra a conexao

    client_connection.close()
```

No código original foi declarado 2 variáveis, para ajudar com o gerenciamento do endereço e foi acrecentado 3 métodos if em cascata para que: caso um método que não fosse get fosse solicitado, um ‘Bad request’ seria exibido; caso o método get fosse digitado (1), exibiria o Index da aplicação (2), ou a página solicitada caso as páginas requeridas existam (3), ou então vai ser exibido um ‘page not found’ (4) no navegador.

Os resultados do código foram satisfatórios na primeira parte, mas não foi possível exibir a página html devido alguns erros de acesso.