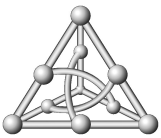


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Trabalho 01: Cliente e Servidor Python

Redes de Computadores

1 Descrição

Este trabalho deve ser entregue no Moodle até a data correspondente de entrega. Envie sua resposta somente em texto a não ser que outro formato seja absolutamente necessário. Sinta-se encorajado para trabalhar em grupo nesta atividade (contudo, sua entrega é individual).

Sua atividade é responder as perguntas em **negrito**.

2 Cliente

Execute o Cliente HTTP Python [1]. Escolha três páginas web e execute o programa para estas páginas. Servidores Web geralmente usam a porta 80 para conexão.

```
#!/usr/bin/python
# A simple http client that retrieves the first page of a web site

import socket, sys

if len(sys.argv)!=3 and len(sys.argv)!=2:
    print "Usage : ",sys.argv[0]," hostname [port]"
hostname = sys.argv[1]
if len(sys.argv)==3 :
    port=int(sys.argv[2])
else:
    port = 80

READBUF=16384
s=None

# size of data read from web server
for res in socket.getaddrinfo(hostname, port, socket.AF_UNSPEC, socket.SOCK_STREAM):
    af, socktype, proto, canonname, sa = res
    # create socket
    try:
        s = socket.socket(af, socktype, proto)
    except socket.error:
        s = None
        continue
    # connect to remote host
    try:
        print "Trying "+sa[0]
        s.connect(sa)
    except socket.error, msg:
```

```
        # socket failed
        s.close()
        s = None
        continue
    if s :
        print "Connected to "+sa[0]
        s.send('GET / HTTP/1.1\r\nHost:'+hostname+'\r\n\r\n')
        finished=False
        count=0
        while not finished:
            data=s.recv(READBUF)
            count=count+1
            if len(data)!=0:
                print repr(data)
            else:
                finished=True
        s.shutdown(socket.SHUT_WR)
        s.close()
        print "Data was received in ",count," recv calls"
        break
```

Listing 1: Cliente HTTP simples em Python.

1. Anote as URLs das páginas que você escolheu e o número de requisições de chamada para obter a página principal de cada página web.

Se você não tem familiaridade com Python, entenda que Python usa a identação para mostrar a estrutura de um programa, logo, tenha certeza que a identação do seu código segue a mesma identação apresentada aqui. Lembrem-se que copiar e colar pode mudar a identação e, nesse caso, você precisará ajustar isso manualmente.

Se você executar este exercício corretamente, deverá ver um cabeçalho de resposta HTTP enviado pelo servidor, assim como o seu conteúdo. Você precisará dessa informação para a parte 6. Algumas páginas enviam devolta uma grande quantidade de dados, você precisará rolar a barra para trás ou procurar uma página web que envie menos dados.

Se você observar com cuidado, notará que o cabeçalho termina com a sequência `\r\n\r\n`, que é um fim de linha seguido por uma linha em branco.

3 Modificando o Cliente

Modifique o cliente da parte 2 para que ele possa requisitar uma URL arbitrária. Você precisará modificar a linha que começa com `s.send` para requisitar um caminho diferente de somente "/". Você precisará também escrever algum código que separe a URL de coisas como:

`http://prof.facom.ufms.br/~brivaldo/pages/project/`

em seus componentes: o endereço do host é `prof.facom.ufms.br` e o caminho requisitado

é `/~brivaldo/pages/project/`. O método `split` de strings pode ser útil. Por exemplo, você pode usar: `parts = url.split("/")`.

2. Modifique e envie seu código. Se seu código funcionar, coloque no seu relatório a quantidade de requisições necessárias para obter: "https://www.ufms.br/cursos/graduacao/".

4 Cliente de Alto Nível

Modifique o segundo cliente 2 para receber uma URL como argumento e salvar o resultado em um arquivo chamado: `dados_web.html` (mesmo que para algumas URLs, os dados não sejam HTML).

```
#!/usr/bin/python
# A simple http client that retrieves the first page of a web site, using
# the standard httplib library

import httplib, sys

if len(sys.argv)!=3 and len(sys.argv)!=2:
    print "Usage : ",sys.argv[0]," hostname [port]"
    sys.exit(1)

path = '/'
hostname = sys.argv[1]
if len(sys.argv)==3 :
    port = int(sys.argv[2])
else:
    port = 80

conn = httplib.HTTPConnection(hostname, port)
conn.request("GET", path)
r = conn.getresponse()
print "Response is %i (%s)" % (r.status, r.reason)
print r.read()
```

Listing 2: Cliente HTTP avançado em Python.

3. Modifique, envie seu código e identifique se ele funciona.

5 Servidor

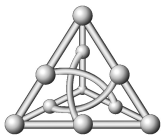
Execute o servidor HTTP simples 3 na sua máquina local e conecte nele usando vários clientes web, incluindo seu próprio código anteior e pelo menos um navegador de Internet normal. Se você estiver executando seu servidor na porta X (por exemplo, na porta 6789), você pode usar a URL `http://localhost:6789` para conectar no servidor usando a mesma máquina.

Agora modifique o servidor para imprimir cada requisição, assim você será capaz de ver que requisições seus clientes estão enviando.

4. Informe o nome do cliente web que você usou para conectar no seu servidor web (Safari, Edge, Firefox, etc.) e determine todas as requisições que o navegador fez ao servidor. Pode ser apenas uma requisição ou podem ser várias. Apenas observe cuidadosamente e informe quais foram as requisições.

```
# An extremely simple HTTP server
import socket, sys, time

# Server runs on all IP addresses by default
HOST=''
# 8080 can be used without root priviledges
PORT=8080
BUFLEN=8192 # buffer size

s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
try:
    print "Starting HTTP server on port ", PORT
    s.bind((HOST,PORT,0,0))
except socket.error :
    print "Cannot bind to port :",PORT
    sys.exit(-1)

s.listen(10) # maximum 10 queued connections

while True:
    # a real server would be multithreaded and would catch exceptions
    conn, addr = s.accept()
    print "Connection from ", addr
    data=''
    while not '\n' in data : # wait until first line has been received
        data = data+conn.recv(BUFLEN)
    if data.startswith('GET'):
        # GET request
        conn.send('HTTP/1.0 404 Not Found\r\n')
        # a real server should serve files
    else:
        # other type of HTTP request
        conn.send('HTTP/1.0 501 Not implemented\r\n')
    now = time.strftime("%a, %d %b %Y %H:%M:%S", time.localtime())
    conn.send('Date: ' + now +'\r\n')
    conn.send('Server: Dummy-HTTP-Server\r\n')
    conn.send('\r\n')
    conn.shutdown(socket.SHUT_RDWR)
    conn.close()
```

Listing 3: Servidor HTTP em Python.



6 Modificando o Servidor

Modifique o servidor para retornar o conteúdo de um único arquivo chamado `index.html`. Você deve criar manualmente este arquivo para testar em conjunto com o seu servidor. Veja um exemplo de um código HTML se você nunca viu um antes:

```
<html>
<head><title>Web Page do Meu Servidor Web</title></head>
<body>Esta &eacute; uma p&aacute;gina Web.</body>
</html>
```

O cabeçalho que o seu servidor retorna deve conter pelo menos algum dos cabeçalhos que vimos na parte 2, especificamente: `Connection: close`, `Server: X` (sendo X o nome que você escolheu para o servidor), `Content-Type: text/html` e `Content-Length: Y` (sendo Y o número de *bytes* obtidos do arquivo `index.html`).

5. Envie seu código do servidor modificado.