

## Atividade – AV2

### Questão 01: Servidor Web

Nesta tarefa, você desenvolverá um servidor Web simples em Python, capaz de processar apenas uma requisição. Seu servidor Web (i) criará um socket de conexão quando contatado por um cliente (navegador); (ii) receberá a requisição HTTP dessa conexão; (iii) analisará a requisição para determinar o arquivo específico sendo requisitado; (iv) obterá o arquivo requisitado do sistema de arquivo do servidor; (v) criará uma mensagem de resposta HTTP consistindo no arquivo requisitado precedido por linhas de cabeçalho; e (vi) enviará a resposta pela conexão TCP ao navegador requisitante. Se um navegador requisitar um arquivo que não está presente no seu servidor, seu servidor deverá retornar uma mensagem de erro "404 Not Found".

A seguir, é oferecido o código estrutural para o seu servidor. Sua tarefa é concluir o código, rodar seu servidor e depois testá-lo enviando requisições de navegadores rodando em hospedeiros diferentes. Se você rodar seu servidor em um hospedeiro que já tem um servidor Web rodando nele, então deverá usar uma porta diferente da porta 80 para o seu servidor.

### Esqueleto de Código Python para o Servidor Web

Abaixo você encontrará o esqueleto do código para o servidor Web. Você deve completar o código onde indicado. Os trechos onde é necessário inserir código estão marcados com **#Fill in start** e **#Fill in end**. Cada um desses trechos pode exigir uma ou mais linhas de código.

```
# Importa o módulo socket from socket import * import sys #
Necessário para encerrar o programa

# Cria o socket TCP (orientado à conexão) serverSocket =
socket(AF_INET, SOCK_STREAM)

# Prepara o socket do servidor

#Fill in start
serverSocket.bind(('', 8000))
ServerSocket.listen(1)
#Fill in end

while True:
    # Estabelece a conexão
    print('Ready to serve...')
    connectionSocket, addr = #Fill in start serverSocket.accept()
                           #Fill in end
    try:
        # Recebe a mensagem do cliente (requisição HTTP)
        #Fill in start
        message = connectionSocket.recv(1024)
```

#Fill in end

```
filename = message.split()[1] f
= open(filename[1:])
outputdata = #Fill in start f.read() #Fill in end

# Envia a linha de status do cabeçalho HTTP
#Fill in start

connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n".encode())

#Fill in end

# Envia o conteúdo do arquivo ao cliente for
i in range(0, len(outputdata)):
    connectionSocket.send(outputdata[i].encode())
connectionSocket.send("\r\n".encode())

# Fecha a conexão com o cliente
connectionSocket.close()

except IOError:
    # Envia mensagem de erro 404 se o arquivo não for encontrado
    #Fill in start
    connectionSocket.send("HTTP/1.1 404 Not
    Found\r\n\r\n".encode())
    connectionSocket.send("<html><body>404                Not
    found<\body></html>".encode())
    #Fill in end

# Fecha o socket do cliente
#Fill in start
connectionSocket.close()
```

#Fill in end

```
serverSocket.close()  
sys.exit() # Encerra o programa
```

## Executando o Servidor

- ❑ Crie um arquivo HelloWorld.html no mesmo diretório do script, com algo simples como: `<html><body><h1>Hello, World!</h1></body></html>`.
- ❑ Execute o servidor
- ❑ No navegador (ou em outro computador da rede), acesse: <http://127.0.0.1:6789/HelloWorld.html> (ou substitua 127.0.0.1 pelo IP da máquina onde o servidor está rodando.)

HelloWorld.html é o nome do arquivo que você colocou no diretório do servidor. Note também o uso do número da porta após os dois pontos. Você deve substituir esse número de porta pelo que você utilizou no código do servidor. Em seguida, tente acessar um arquivo que não existe no servidor. Você deverá receber uma mensagem “404 Not Found”.

## Questão 02: Wireshark – HTTP

### 1. O Básico do GET do HTTP/interação de resposta

Vamos começar explorando o HTTP fazendo o download de um arquivo HTML muito simples – um arquivo pequeno e que não contém referências para objetos. Faça o seguinte:

- ❑ Inicie o navegador Web.
- ❑ Inicie o sniffer de pacotes, Wireshark, mas não inicie a captura de pacotes. Digite “http” (somente as letras sem as aspas) na janela display-filter-specification, assim somente as mensagens HTTP serão mostradas pela ferramenta.
- ❑ Espere um instante antes de iniciar a captura de pacotes.

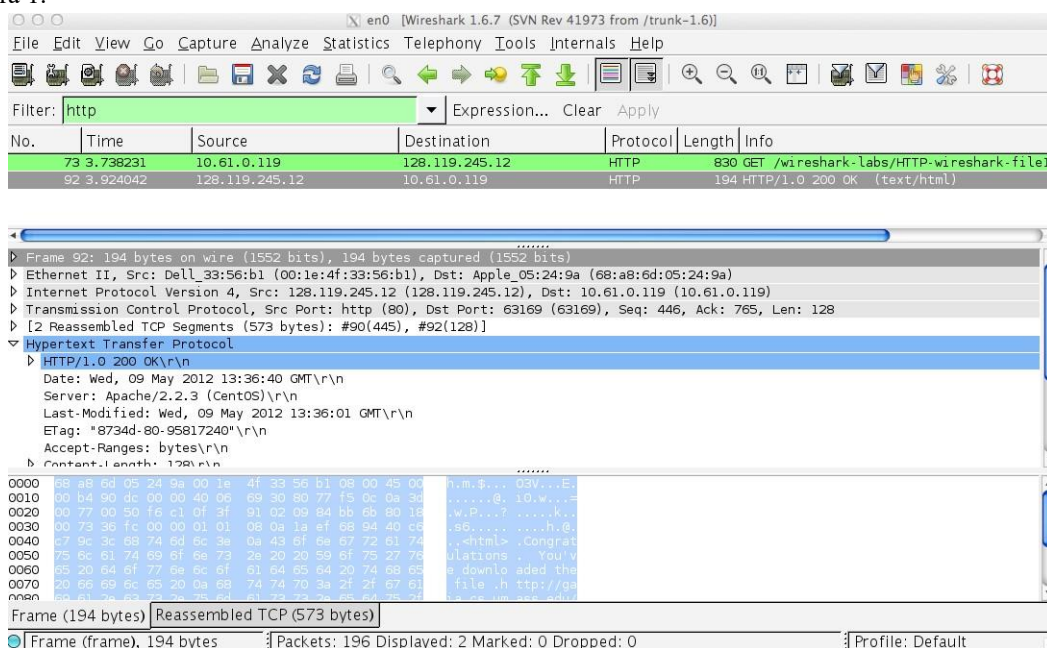
- Entre o endereço abaixo no navegador: <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>

O navegador deve mostrar uma página bem simples com uma linha.

- Pare a captura de pacotes do Wireshark.

A sua janela do Wireshark deve ser bem parecida com a mostrada na Figura 1. Se não estiver apto a rodar o Wireshark em uma conexão de rede, você pode fazer o download do pacote que foi criado, quando os passos acima foram executados<sup>2</sup>.

<sup>2</sup>Baixe o arquivo zip <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip> e extraia o arquivo http-ethereal-trace-1. O arquivo zip foi coletado quando o autor do livro rodou o Wireshark enquanto executava os passos para o lab. Após baixar o arquivo, você pode abrir no Wireshark e ver as informações usando File -> Open e selecionando o arquivo http-ethereal-trace-1 trace. A tela resultante deve ser parecida com a Figura 1.



**Figura 1.** Tela do Wireshark depois de entrar em <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>

O exemplo na Figura 1 mostra na janela de listagem duas mensagens HTTP capturadas: uma mensagem de GET (de um navegador para o servidor [gaia.cs.umass.edu](http://gaia.cs.umass.edu)) e uma mensagem de resposta do servidor para o navegador. A janela com o conteúdo dos pacotes mostra detalhes da mensagem selecionada. Lembre-se de que a mensagem HTTP é carregada num segmento TCP, o qual está num datagrama IP, o qual está num quadro Ethernet. O Wireshark mostra informações do quadro, Ethernet, IP e TCP. Queremos minimizar a quantidade de dados que não são HTTP, então mostre apenas as informações de HTTP no Wireshark e esconda o restante.

Observando a informação do GET HTTP e a mensagem de resposta, responda as questões abaixo. Quando responder, inclua uma impressão das mensagens de GET e a resposta, e indicar onde você encontrou as respostas.

- a. O seu navegador está rodando a versão 1.0 ou 1.1 do HTTP? Que versão do HTTP está sendo rodada no servidor?

A versão do HTTP que esta rodando é a 1.1.

**b.** Que linguagem (se tiver) o seu navegador indica que ele pode aceitar do servidor?

Não possui linguagem.

**c.** Qual é o endereço IP do seu computador? E do servidor `gaia.cs.umass.edu`?

92.168.0.139 – Meu computador.

128.119.245.12 – Servidor

**d.** Qual é o código de estado retornado pelo servidor para o seu navegador?

200 – OK

**e.** Quando foi a última alteração feita no arquivo HTML?

17/11/2025

**f.** Quantos bytes de conteúdo são retornados para o seu navegador?

543 bytes

## **2. GET HTTP CONDITIONAL/interação de resposta**

Antes de fazer esta parte, certifique-se de que o cache do navegador esteja limpo (faça isso usando a opção para apagar o histórico do navegador). Agora, faça o seguinte:

- ☐ Inicie o navegador e certifique-se de que o cache esteja limpo.
- ☐ Inicie o Wireshark.
- ☐ Entre com a URL no navegador `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html`  
Seu navegador deve mostrar um arquivo bem simples HTML com cinco linhas.
- ☐ Rapidamente entre com a URL no navegador novamente (ou simplesmente selecione o botão para recarregar a página).
- ☐ Pare o Wireshark e entre com “http” na janela em que se especifica filtros para que somente as mensagens HTTP possam ser mostradas na janela de listagem de pacotes.
- ☐ (Nota: Se você não puder rodar o Wireshark em uma conexão interativamente, use o disposto no rodapé 2 para responder as perguntas abaixo)

Responda às seguintes questões:

**g.** Inspeção o conteúdo da primeira requisição GET do HTTP de seu navegador para o servidor. Você viu uma linha com “IF-MODIFIED-SINCE” no GET do HTTP?

Não

**h.** Inspeção o conteúdo da resposta do servidor. O servidor retornou o conteúdo do arquivo explicitamente? Como isso se observou?

Sim, indo no souce de inspeção de elemento do navegador.

i. Agora investigue o conteúdo da segunda requisição GET do HTTP de seu navegador para o servidor. Você viu uma linha com “IF-MODIFIED-SINCE” no GET do HTTP? Se sim, que informação segue o cabeçalho “IF-MODIFIED-SINCE:”?

Não

j. Qual é o código de estado do HTTP e a frase retornada do servidor em resposta ao segundo GET do HTTP? O servidor explicitamente retornou o conteúdo do arquivo? Explique

Not found, não ele retornou o erro 404

The image shows a Wireshark packet capture of an HTTP session. The packet list at the top shows several packets, with the last one (No. 950) being an HTTP 200 OK response. The packet details pane on the right shows the structure of the HTTP response, including the status bar '200 OK'.

No.	Time	Source	Destination	Protocol	Length	Info
671	29.689127	192.168.0.139	192.168.0.1	HTTP/X...	689	POST /uuid:d41e4672-
677	29.767767	192.168.0.1	192.168.0.139	HTTP/X...	415	HTTP/1.1 200 OK
685	29.782625	192.168.0.139	192.168.0.1	HTTP/X...	992	POST /uuid:d41e4672-
690	29.890747	192.168.0.1	192.168.0.139	HTTP/X...	343	HTTP/1.1 200 OK
697	29.901460	192.168.0.139	192.168.0.1	HTTP/X...	992	POST /uuid:d41e4672-
704	30.015585	192.168.0.1	192.168.0.139	HTTP/X...	343	HTTP/1.1 200 OK
710	30.028139	192.168.0.139	192.168.0.1	HTTP/X...	992	POST /uuid:d41e4672-
715	30.140100	192.168.0.1	192.168.0.139	HTTP/X...	343	HTTP/1.1 200 OK
722	30.153799	192.168.0.139	192.168.0.1	HTTP/X...	992	POST /uuid:d41e4672-
727	30.260818	192.168.0.1	192.168.0.139	HTTP/X...	343	HTTP/1.1 200 OK
947	42.872166	192.168.0.139	192.168.0.141	HTTP	279	GET / HTTP/1.1
950	42.903400	192.168.0.141	192.168.0.139	HTTP/X...	1279	HTTP/1.1 200 OK

Frame 531: Packet, 222 bytes on wire (1776 bits),  
Ethernet II, Src: TpLinkTechno\_15:38:de (7c:8b:ca:00:10:00), Dst: 192.168.0.139 (08:00:27:00:00:00)  
Internet Protocol Version 4, Src: 192.168.0.139, Dst: 192.168.0.1  
Transmission Control Protocol, Src Port: 49508, Dst Port: 80  
Hypertext Transfer Protocol

+	400	1.135450	192.168.0.139	192.168.0.136	HTTP	239	GET /ssdp/device-desc.xml	
+	403	1.151007	192.168.0.136	192.168.0.139	HTTP/X...	1253	HTTP/1.1 200 OK	
	442	1.257543	192.168.0.139	192.168.0.136	HTTP	248	GET /apps/com.spotify.Spoti	
	446	1.274334	192.168.0.136	192.168.0.139	HTTP	98	HTTP/1.1 404 Not Found	
	15090	52.271907	192.168.0.139	128.119.245.12	HTTP	573	GET /wireshark-labs/HTTP-w	
	15111	52.405290	128.119.245.12	192.168.0.139	HTTP	543	HTTP/1.1 200 OK (text/htm	
	15129	52.466232	192.168.0.139	128.119.245.12	HTTP	519	GET /favicon.ico HTTP/1.1	
	15152	52.598867	128.119.245.12	192.168.0.139	HTTP	636	HTTP/1.1 301 Moved Permanen	

▶	Frame 400: Packet, 239 bytes on wire (1912 bits),	0000	46 98 66 7d 3c cc 7c 8b ca 15 38 de 08 00 4	▶
▶	Ethernet II, Src: TpLinkTechno_15:38:de (7c:8b:ca:	0010	00 e1 34 d8 40 00 80 06 42 db c0 a8 00 8b c	
▶	Internet Protocol Version 4, Src: 192.168.0.139, D	0020	00 88 e1 b5 1f 48 bf 09 51 59 97 9a af 7a 5	
▶	Transmission Control Protocol, Src Port: 57781, Ds	0030	00 ff 0f 3d 00 00 47 45 54 20 2f 73 73 64 7	
▶	Hypertext Transfer Protocol	0040	64 65 76 69 63 65 2d 64 65 73 63 2e 78 6d 6	
		0050	48 54 54 50 2f 31 2e 31 0d 0a 55 73 65 72 2	
		0060	67 65 6e 74 3a 20 53 70 6f 74 69 66 79 2f 3	
		0070	37 35 30 30 35 31 30 20 57 69 6e 33 32 5f 7	
		0080	36 5f 36 34 2f 30 20 28 50 43 20 64 65 73 6	
		0090	6f 70 29 0d 0a 48 6f 73 74 3a 20 31 39 32 2	
		00a0	36 38 2e 30 2e 31 33 36 3a 38 30 30 38 0d 0	
		00b0	65 65 70 2d 41 6c 69 76 65 3a 20 30 0d 0a 4	
		00c0	63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 2	
		00d0	7a 69 70 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6	

Hypertext Transfer Protocol: Protocol Pacotes: 17910 · Exibidos: 8 (0.0%) Perfil: Default