Laboratório 2: Servidores Web

1. Objetivo

Pretende-se com a realização deste exercício de laboratório que o aluno aprenda os fundamentos da programação de *sockets* para ligações TCP em Python. Mais especificamente:

- como criar um *socket*, ligá-lo a um endereço e porta específicos, bem como enviar e receber um pacote HTTP;
- relembrar e aplicar algumas noções básicas sobre o formato do cabeçalho HTTP.

2. Descrição do exercício

O servidor Web, cujo código (Python) encontra-se em anexo, manipula um pedido HTTP de cada vez. O servidor Web aceita e analisa o pedido; procura e obtém o ficheiro solicitado no seu sistema de ficheiros. Depois, compõe uma mensagem de resposta HTTP consistindo no ficheiro solicitado, precedido por linhas de cabeçalho e, em seguida, envia essa resposta ao cliente. Se o ficheiro solicitado não estiver presente no servidor, a resposta ao cliente com a mensagem: HTTP *"404 Not Found "*

3. Execução do servidor

Para executar o processo servidor:

- Coloque um ficheiro HTML (por exemplo, ArqRedes.html) no mesmo diretório em que se encontra o código do servidor;
- Execute o servidor.

4. Acesso ao servidor

Para aceder a uma página do servidor Web através de um navegador (*browser*), execute os seguintes passos:

- Determine o endereço IP do *host* que está executando o servidor (por exemplo: 192.168.1.207);
- Abra um navegador e forneça o URL correspondente. (por exemplo: *http://192.168.1.207/ArqRedes.html*)
    - **OBS:** Se estiver a utilizar o mesmo *host* que executa o servidor poderá também fornecer o seguinte URL: http://localhost/ArqRedes.html (ou ainda http://127.0.0.1/ArqRedes.html)

Para receber a mensagem de *"404 Not Found"* tente aceder a uma página não existente no servidor (por exemplo, *index.html* ou outra qualquer)

5. Nota

Poderá alterar o código do servidor Web para este responder ao pedido de páginas num porto diferente do número 80 (por omissão), como por exemplo o 6789, ou outro valor superior ao número 1024. Neste caso no navegador terá que fornecer esse mesmo número logo assim ao IP separados pelo caracter ":" (dois pontos), como por exemplo: *http://192.168.1.207**:6789**/ArqRedes.html* (ou, se for o caso, http://localhost**:6789**/ArqRedes.html)

6. Exercício A

O servidor Web disponibilizado lida com apenas um pedido HTTP de cada vez. Pretende-se que altere o respetivo código para que passe a ser um servidor *multithread* capaz de atender vários pedidos simultaneamente.

(Por exemplo, criando primeiro uma thread principal para que o servidor escute pedido de clientes num porto fixo, como por exemplo o 80, de modo a que quando receber um pedido de ligação TCP, essa thread principal irá configurar essa ligação através de um outro porto para responder ao pedido desse mesmo cliente. Haverá, assim, uma ligação TCP separada em uma thread separada para cada pedido/resposta)

7. Exercício B

Ao invés de usar um web browser, implemente um cliente HTTP para testar o servidor. O cliente ligar-se-á ao servidor através do protocolo TCP para enviar um pedido HTTP e receber a resposta do servidor que será mostrada no seu output. Pode-se assumir que o cliente faça pedidos com o método GET.

O cliente faz os pedidos através da linha de comandos com argumentos que especificam o endereço IP, ou o *hostname*, do servidor, o número do porto de escuta do servidor e o nome do ficheiro solicitado.

(Por exemplo: `client.py server_host server_port filename`)

8. Referências

- Computer Networking A Top Down Approach , 7th Edition , Kurose….
- Redes de Computadores e Iternet, 6ª Edição, Kurose…