Computer Networks
Season 2024-I
Workshop No. 4 — Sockets and Services

Eng. Carlos Andrés Sierra, M.Sc.
Computer Engineering
Universidad Distrital Francisco José de Caldas

You made a good job in packet tracer. So, it means you understand how to use the tool and how to setup a simple network using concepts as *IP Addressing*, *Cables*, *Routing*, and *Services*. Now, you are going to learn about *Sockets* and *Services*, so you should start on your last week work and add a couple of things.

The main goal of this workshop is to use sockets and determine how OSI model works here:

1. Your are the new computer engineer at *Universidad Distrital Francisco José de Caldas*. You need to increase your *backend* network services. As follows some indications about it.

2. Create a new server. This server should have the following characteristics:

   (a) Have be recognized by the name `BackendPython`.

   (b) Have a public statis IP address, and a default gateway. In this sense, next values should be used:

   - IPv4 Address: `193.168.100.201`
   - Default Gateway: `193.168.100.1`
   - Subnet Mask: `255.255.255.0`

   (c) In the *Programming* section add a new **Project** with the name of your preference, but it must be ***Template**: Empty - Python*. In the `main.py` file add the following code:

```
"""
This is a simple example of a web service for Python into PacketTracer.

Author: Carlos Andres Sierra <cavirguezs@udistrital.edu.co>
```

Carlos Andrés Sierra, Computer Engineer, M.Sc. on Computer Engineering, Titular Professor at Universidad Distrital Francisco José de Caldas.

Any comment or concern related to this document could be send to Carlos A. Sierra at e-mail: *cavirguezs@udistrital.edu.co*

```
"""

from http import *
from time import *

def on_route_networks(url: str, response):
    """
    This function is called when the URL is /healthcheck.

    Args:
        url (str): The URL of the request.
        response (HTTPResponse): The response object to send data to the client.
    """
    print("Test Services")
    response.send("This is a verification about python services.")

def main():
    """This is the main function of the program."""
    HTTPServer.route("/healthcheck", on_route_networks)
    # start server on port 80
    print(HTTPServer.start(80))
    # don't let it finish
    while True:
        sleep(3600)

if __name__ == "__main__":
    main()
```

3. Create another new server. This server should have the following characteristics:

   (a) Have be recognized by the name `BackendJavaScript`.

   (b) Have a public statis IP address, and a default gateway. In this sense, next values should be used:

   - IPv4 Address: `193.168.100.202`
   - Default Gateway: `193.168.100.1`
   - Subnet Mask: `255.255.255.0`

   (c) In the *Programming* section add a new **Project** with the name of your preference, but it must be ***Template**: Empty - Javascript*. In the `main.js` file add the following code:

```
/*
This is a simple example of a web service for Python into PacketTracer.

Author: Carlos Andres Sierra <cavirguezs@udistrital.edu.co>
```

```
*/

function setup() {
    HTTPServer.route("/healthcheck", function(url, res) {
        Serial.println("Test services");
        res.setContentType("text/plain");
        res.send("This is a verification about javascript services");
    });

    // start server on port 80
    HTTPServer.start(80);
}
```

4. The frontend server, same you build next with, should change a little bit. Be creative, improve the *Look & Feel*, even you could create better services. It is part of your work too.

   (a) You could use next *HTML* code in a `index.html` file to create a simple web page:

```
<html>
    <head>
        <title>Workshop 4 - Networks</title>
        <meta charset="UTF-8">
        <meta name="description" content="This is a simple example to explore OSI
        <link rel="stylesheet" type="text/css" href="styles.css">
        <script src="functions.js"></script>
    </head>
    <body>
        <h1>Workshop 4 - Networks</h1>
        <h2>OSI Layers</h2>
        <p>Click on the buttons to explore services on the network</p>
        <div>
            <button onclick="callPython()">Python Message</button>
            <button onclick="callJavaScript()">JavaScript Message</button>
        </div>
        <div id="result"></div>
    </body>
</html>
```

   (b) You could use next *CSS* code in a `styles.css` file to create a simple style for the web page:

```
body {
    font-family: "Arial", sans-serif;
    font-size: 16px;
```

```
}

h1 {
    color: #ea0909;
    font-size: 24px;
    margin-bottom: 20px;
}

h2 {
    color: #333;
    font-size: 20px;
    margin-bottom: 10px;
}

button {
    background-color: #e6ea09;
    color: rgba(215, 21, 21, 0.697);
    border: none;
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    border-radius: 10px;
}

.result {
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #333;
    border-radius: 10px;
}
```

(c) You could use next *JavaScript* code in a `functions.js` file to create a simple style for the web page:

```
function callPython() {
    fetch('http://193.168.100.201/healthcheck')
        .then(response => response.text())
        .then(data => {
            const resultDiv = document.getElementById('result');
            resultDiv.innerText = data;
        })
        .catch(error => {
            console.error('Error:', error);
        });
}
```

```
function callJavaScript() {
    fetch('http://193.168.100.202/healthcheck')
        .then(response => response.text())
        .then(data => {
            const resultDiv = document.getElementById('result');
            resultDiv.innerText = data;
        })
        .catch(error => {
            console.error('Error:', error);
        });
}
```

To test the network, you need to access to a web browser in the `StudentLaptop` and type the URL `www.udistrital.edu.co`. Same test should be done in the `WorkerPC`.

The output should be the *university home page* you created into the server with the *buttons* to call the web services

As final result of this workshop you should deliver a *PDF* file with and screenshot of the network, the decisions you make to build the networks, and you analysis using *Simulation* option in *PacketTracer* to validate how the network works across the *OSI Model*.

Bonus: Create a `VLAN` where servers *BackendPython* and *BackendJavaScript* are accessible just for *FrontEnd Server*, but not from laptop or pc on the other side of the network.