Proyecto TICS: Diseño de Red

Diego Irarrázaval

22 de diciembre de 2021

Enlaces del proyecto:

- Videos: https://drive.google.com/drive/folders/1IL9Yv6fpkiTOQzF5NWlQ50Yl6rgvDSKM?usp=sharing
- Protecto en Github: https://github.com/Diego-II/Proyecto-TICS/

1. Diseño Físico

El diseño físico que se muestra a continuación es una simplificación del diseño físico que se intentó implementar en primera instancia (ver anexo Diseño físico inicial). En el implementado, se eliminan los switches que separan **LANs** en cada zona o ciudad y se mantiene simplemente dos **PCs** conectados directamente a un switch:

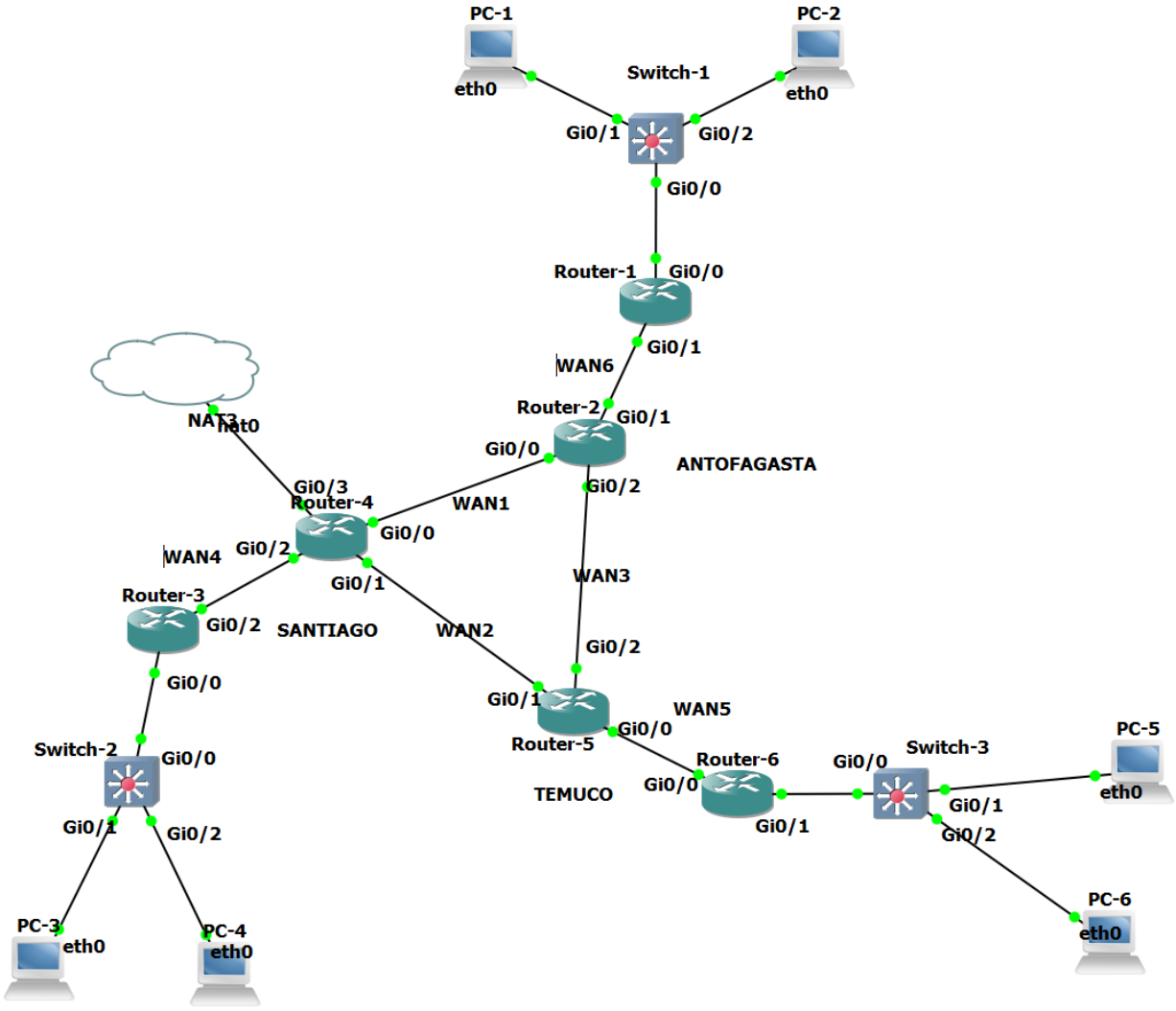

Figura 1: Diseño físico de la red.

Para describir de forma detallada cada red, procedemos al diseño lógico de ésta.

2. Diseño Lógico

RED	Subnet address	Netmask	Range of addresses	Useable IPs	Hosts
LAN1	222.4.0.0/26	255.255.255.192	222.4.0.0 - 222.4.0.63	222.4.0.1 - 222.4.0.62	62
LAN2	222.4.0.64/27	255.255.255.224	222.4.0.64 - 222.4.0.95	222.4.0.65 - 222.4.0.94	30
LAN3	222.4.0.96/27	255.255.255.224	222.4.0.96 - 222.4.0.127	222.4.0.97 - 222.4.0.126	30
LAN4	222.4.0.128/29	255.255.255.248	222.4.0.128 - 222.4.0.135	222.4.0.129 - 222.4.0.134	6
LAN5	222.4.0.136/29	255.255.255.248	222.4.0.136 - 222.4.0.143	222.4.0.137 - 222.4.0.142	6
LAN6	222.4.0.144/29	255.255.255.248	222.4.0.144 - 222.4.0.151	222.4.0.145 - 222.4.0.150	6
WAN1	222.4.0.152/30	255.255.255.252	222.4.0.152 - 222.4.0.155	222.4.0.153 - 222.4.0.154	2
WAN2	222.4.0.156/30	255.255.255.252	222.4.0.156 - 222.4.0.159	222.4.0.157 - 222.4.0.158	2
WAN3	222.4.0.160/30	255.255.255.252	222.4.0.160 - 222.4.0.163	222.4.0.161 - 222.4.0.162	2
WAN4	222.4.0.168/30	255.255.255.252	222.4.0.168 - 222.4.0.171	222.4.0.169 - 222.4.0.170	2
WAN5	222.4.0.172/30	255.255.255.252	222.4.0.172 - 222.4.0.175	222.4.0.173 - 222.4.0.174	2
WAN6	222.4.0.176/30	255.255.255.252	222.4.0.176 - 222.4.0.179	222.4.0.177 - 222.4.0.178	2

Cuadro 1: Diseño lógico de la red.

En primer lugar, es importante mencionar que la red que debiese ir como **WAN4** (correspondiente a las direcciones `222.4.0.164 - 222.4.0.167`), no está en la tabla debido a que originalmente, se contemplaba otra red más con prefijo `/30` correspondiente a la red entre el router 4 y **NAT**, pero esto durante la implementación se notó que no era necesario.

En la tabla 2, se presenta el detalle de cada red, con una breve descripción de cada una, el rango de dirección y el **gateway** en aquellas que aplique.

RED	Descripción	Gateway (si aplica)	Range of addresses	Hosts Solicitados
LAN1	Santiago - PCs	222.4.0.1	222.4.0.0 - 222.4.0.63	36
LAN2	Temuco - PC's	222.4.0.65	222.4.0.64 - 222.4.0.95	30
LAN3	Antofagasta - PC's	222.4.0.97	222.4.0.96 - 222.4.0.127	30
LAN4	Santiago - Sensores	222.4.0.1	222.4.0.128 - 222.4.0.135	6
LAN5	Temuco - Sensores	222.4.0.65	222.4.0.136 - 222.4.0.143	6
LAN6	Antofagasta - Sensores	222.4.0.97	222.4.0.144 - 222.4.0.151	6
WAN1	Stgo - Ant		222.4.0.152 - 222.4.0.155	2
WAN2	Stgo - Temuco		222.4.0.156 - 222.4.0.159	2
WAN3	Ant - Temuco		222.4.0.160 - 222.4.0.163	2
WAN4	Stgo: R3 - R4		222.4.0.168 - 222.4.0.171	2
WAN5	Temuco: R6 - R5		222.4.0.172 - 222.4.0.175	2
WAN6	Antofagasta: R1 - R2		222.4.0.176 - 222.4.0.179	2

Cuadro 2: Diseño lógico de la red con **gateway** en casos que aplique y breve descripción de cada una.

Respecto a la asignación eficiente de direcciones IP, en el único caso en el que no se cumple es el mencionado anteriormente respecto a la red **R4 - NAT** que no se implementó.

3. Python y paquetes utilizados

Para la instalación de python y paquetes era necesario que las terminales de ubuntu docker tuviesen conexión a internet, lo que no se logró nunca. Pero, los pasos para lograr la correcta instalación son los siguientes:

```
apt-get update
apt-get install software-properties-common
apt-get update
apt-get install python3.8
apt-get -y install python3-pip
```

Algoritmo 1: Instrucciones para instalación de Python.

Adicionalmente, se debe instalar la librería `ubidots` con `pip3 install -r requirements.txt`

Para subir el código y que esté disponible en la máquina Ubuntu, este se subio a Github con la finalidad de ser clonado y asi obtener todos los archivos necesarios: https://github.com/Diego-II/Proyecto-TICS/

4. Scripts de Python

```
from logging import error
from ubidots import ApiClient
import dotenv
import os
```

```
import requests
import random
import time

def check_variables_entorno():
    dotenv.load_dotenv(dotenv_path='.env')
    try:
        UBIDOTS_API_TOKEN = os.environ['UBIDOTS_API_TOKEN']
        UBIDOTS_DEVICE_ID = os.environ['UBIDOTS_DEVICE_ID']
    except:
        raise error("Variables de ubidots no configuradas en .env")

def get_client_ubidots(token):
    return ApiClient(token='BBFF-mfYqpyZM2LBYAdABfUwrucY4XNOlCw')

def get_variable(api_client: ApiClient, variable):
    return api_client.get_variable('60b44d4f73efc32af6b4e3ac')

def load_data(file_path: str):
    """entrega arreglo con los datos leidos del archivo en csv. En la realidad, leeria los datos del
    sensor.

    Args:
        file_path (str): Path al archivo .csv
    """
    import csv
    with open(file_path, newline='') as csvfile:
        datos = []
        for fila in csv.reader(csvfile):
            datos.append(int(fila[0]))
    return datos

def post_var(sensor_value, url, device, variable, token):
    """
    Funcion que envia una solicitud POST a la api de ubidots subiendo el valor
    'sensor_value' segun las credenciales y llaves indicadas en
    url, device y token.
    """
    try:
        url = "http://{}/api/v1.6/devices/{}".format(url, device)
        headers = {"X-Auth-Token": token, "Content-Type": "application/json"}

        attempts = 0
        status_code = 400

        payload = {variable: sensor_value}

        while status_code >= 400 and attempts < 5:
            print("[INFO] Sending data, attempt number: {}".format(attempts))
            req = requests.post(url=url, headers=headers,
```

```
                                json=payload)
            status_code = req.status_code
            attempts += 1
            time.sleep(1)

        print("[INFO] Results:")
        print(req.text)
    except Exception as e:
        print("[ERROR] Error posting, details: {}".format(e))


def main():
    check_variables_entorno()

    ENDPOINT = "app.ubidots.com"
    DEVICE_LABEL = "lab2-iot"
    VARIABLE_LABEL = "ph"
    datos = load_data("datos.csv")
    for i, dato in enumerate(datos[:3]):
    # OJO: Ubidots trabaja con milisegundos de timestamp!
        payload = {"value": dato, "timestamp": int(time.time() - 60 * 60 * (len(datos) - i))*1000}
        post_var(payload, url=ENDPOINT, device=DEVICE_LABEL, variable=VARIABLE_LABEL,token=os.environ['
    UBIDOTS_API_TOKEN'])
    print("subida exitosa!")


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

Algoritmo 2: Script para subida de datos de sensor.

Anexo

Diseño físico inicial

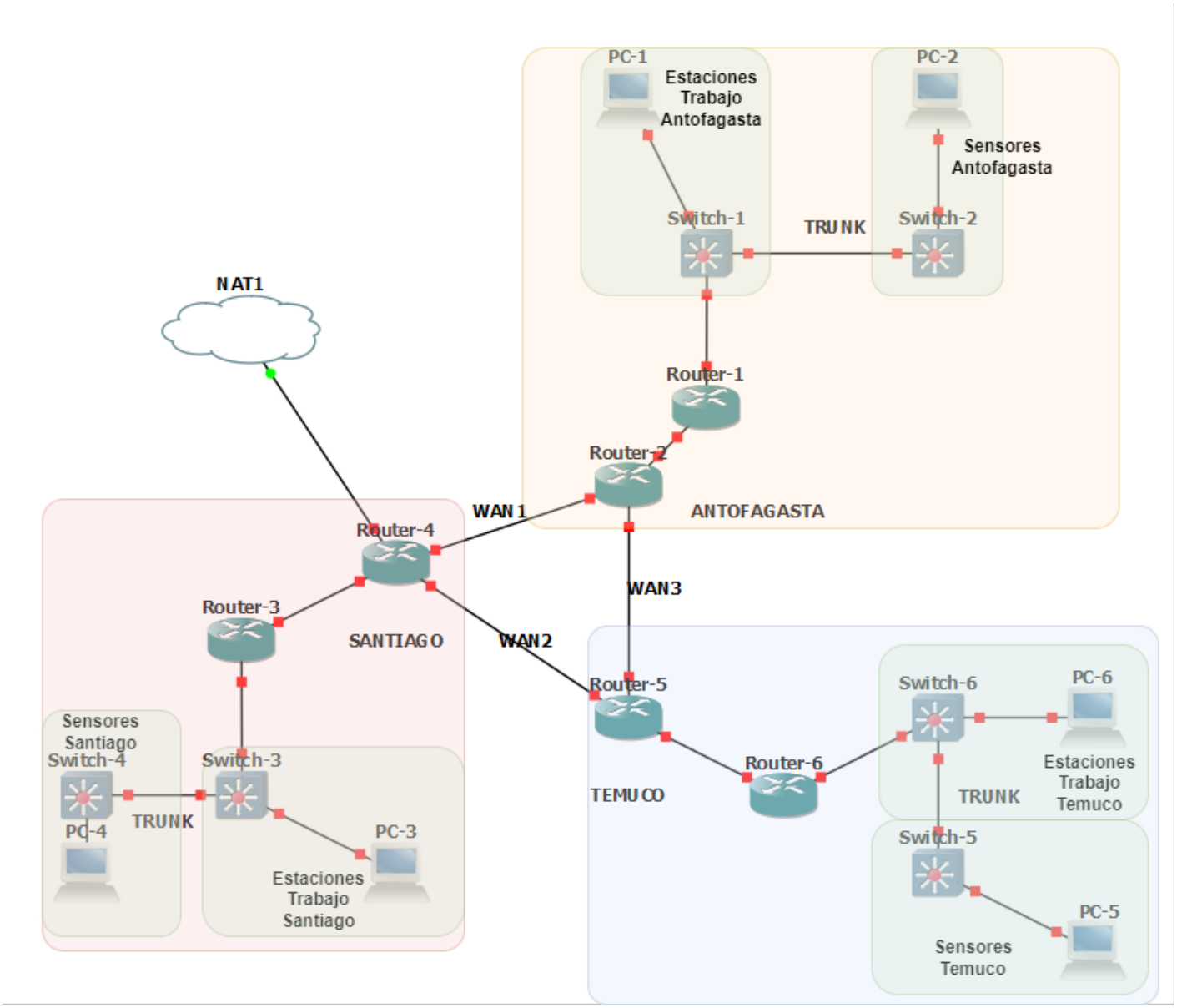

Figura 2: Diseño físico de la red.