UNIVERSIDAD DE GRANADA

Grado en Ingeniería de Tecnologías de Telecomunicación

LABORATORIO DE TELEMATICA

UNIVERSIDAD DE GRANADA

"DESARROLLO DE SOFTWARE PARA EL DISEÑO DE REDES VoIP"

PRACTICA 2

Autor :
José Luis Tejero López
Jorge Suarez Diaz
Martin Torres Antunez

GRANADA, 2021

Índice

1. Introducción

Esta práctica ha sido un desarrollo de un software que nos permita hacer el cálculo de todos los diferentes parámetros relacionados con las redes de VoIP. De esta forma seleccionando los requisitos de la red y los niveles de calidad esperados podemos calcular el ancho de banda necesario o los clientes soportados para una probabilida de bloqueo dada.

Todo el código asociado a esta software esta disponible en el siguiente repositorio:

`https://github.com/joseluistejero/voipNetworkQoS`

2. Instalación del software

2.1. Requisitos de instalación

Previo a la instalación del software es necesario que el equipo donde se va a instalar cumpla con los requisitos técnicos que permitan la ejecución del software. En nuestro proyecto estos son los requisitos:

- Versión 3 o superior de python
- Disponible en los sistemas operativos: Windows, Linux y MACos

El software puede ser descargado directamente desde el repositorio github en formato ZIP o bien desde la linea de comandos con el siguiente comando:

```
git clone https://github.com/joseluistejero/voipNetworkQoS
```

3. Código desarrollado

Para llevar a cabo el desarrollo del software que nos permita calcular que codecs son los que mejor se ajustan a nuestra red dado unos requisitos de entrada hemos hecho uso de los siguientes parámetros que nos permiten caracterizar de manera muy precisa nuestra red.

Todos los parámetros han sido limitados en la interfaz al tipo de dato y rango que hemos considerado adecuado para cada uno de ellos. Si se desea cambiar es posible modificando la interfaz desde QT creator.

Estos han sido estructurados en 5 bloques:

1. Parámetros de llamada:
    - Número de clientes que queremos dimensionar en nuestra empresa
    - Número de lineas por cada cliente
    - Tiempo medio de llamada
2. Parámetros de calidad de experiencia:
    - MOS (Mean Opinion Score): asocia una opinión subjetiva de la calidad de llamada con los distintos codecs

3. Parámetros de calidad de servicio:
    - Retardo total: dependiendo de las necesidades de la aplicación el retardo puede estar entre 0-150ms o entre 150ms y 400ms
    - El retardo de red: en función de datos previos de uso de la red
    - TcWAN: en función del protocolo usado para encapsular entre nuestra empresa y el proveedor VoIP
4. Parámetros de grado de servicio:
    - Protocolo Ethernet usado (802.1, 802.1q, 802.1qinq)
    - Protocolo de capa de enlace (PPP, PPPoE)
    - Uso de túneles (IPSEC, MPLS, MPLS)
5. Parámetros de ancho de banda:
    - Probabilidad de bloqueo: definde la probabilidad de que el cliente no pueda cursar una llamada
    - Ancho de banda de reserva: porcentaje de ancho de banda que queremos reservar para no congestionar la red al máximo en el peor de los casos.
    - Fichero de 0 y 1 que define el número de paquetes exitosos (0) y perdidas (1) para el calculo de ráfagas y probabilidad de perdida.

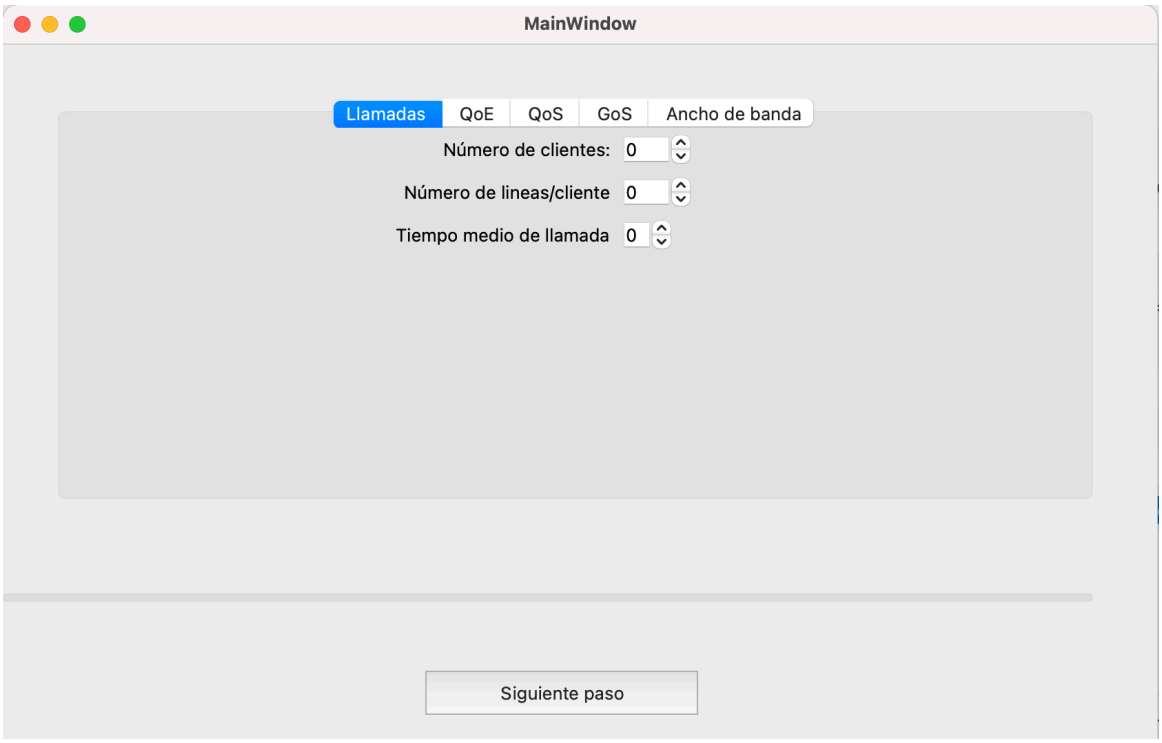

Figura 1: Captura de interfaz de entrada de parámetros

3.1. Estructuración del código

El código ha sido desarrollado en 6 archivos python principales que representan distintas clases. Estos **6 archivos** aparecen cada uno de ellos como un bloque diferente en la siguiente figura. Además se han definido **6 clases** diferentes incluyendo dos de ellas en un mismo archivo. A continuación se detallarán más en profundidad cada uno de estos bloques, su cometido y el contenido de ellos.

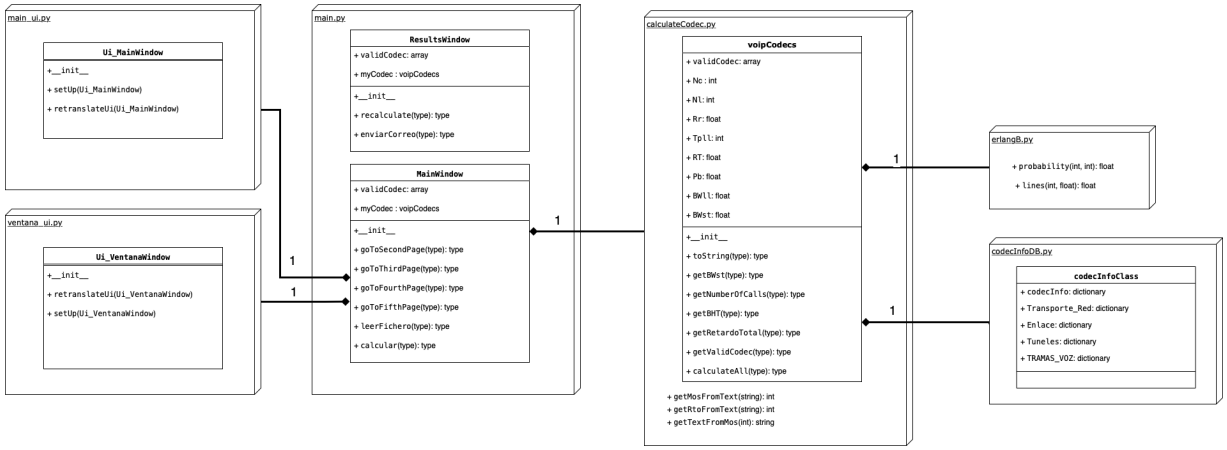

Figura 2: Diagrama de bloques del código

3.1.1. erlangB.py

Este archivo contiene 2 funciones principales relacionadas con la formula que relaciona la probabilidad de bloqueo, el número de lineas y el tráfico BHT de erlangB. Ha sido extraído del siguiente repositorio de github: `https://github.com/diegodafranca/erlangB`. Las funciones son las siguientes:

- probability: calcular la probabilidad de bloqueo dado un número de lineas y una cantidad de tráfico.
- lines: calcular el número de lineas para una probabilidad de bloque y una cantidad de tráfico dada.

3.1.2. codecInfoDB.py

Este archivo contiene una clase llamada codecInfoClass. Esta clase ha sido diseñada como base de datos donde almacenar todos los codecs con sus respectivos parámetros como MOS, CSS, CSI, VPS, etc… Con ello conseguimos modularizar el código a la vez que se hace más sencillo el añadir más codecs.

Contiene 4 diccionarios de python:

- **codecList**: es un diccionario anidado que recoge todos los codecs y sus parámetros asociados
- **Transporte_Red:** es un diccionario que recoge el valor de las cabeceras de los distintos protocolos de red que pueden usarse
- **Enlace**: es un diccionario que recoge el valor de las cabeceras de los distintos protocolos de enlace (Ethernet (802.1, 802.1q, 802.1qinq))
- **Túneles**: es un diccionario que recoge el valor de las cabeceras de los distintos protocolos de tunel (IPsec, MPLS, L2TP)

3.1.3. main_ui.py

Este archivo contiene una clase que representa a la interfaz principal de nuestro software. Esta se ha definido como Ui_MainWindow. Su creación se ha llevado a cabo de manera automática gracias a la herramienta **QT creator** que nos permite diseñar la interfaz de manera gráfica y posteriormente convertir este diseño a código python. Para ello es necesario el uso del siguiente comando:

```
exec python -m PyQt5.uic.pyuic main_iu.ui -o main_ui.py  -x
```

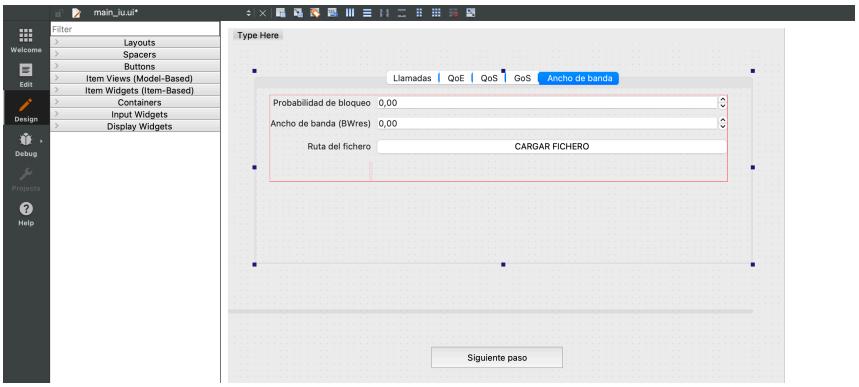

Figura 3: Interfaz de diseño QT creator

3.1.4. ventana_ui.py

Este archivo contiene una clase que representa a la interfaz de resultados de nuestro software. Esta se ha definido como Ui_VentanaWindow. Su creación se ha llevado a cabo de manera automática gracias a la herramienta **QT creator** que nos permite diseñar la interfaz de manera gráfica y posteriormente convertir este diseño a código python. Para ello es necesario el uso del siguiente comando:

```
exec python -m PyQt5.uic.pyuic ventana_iu.ui -o ventana_ui.py  -x
```

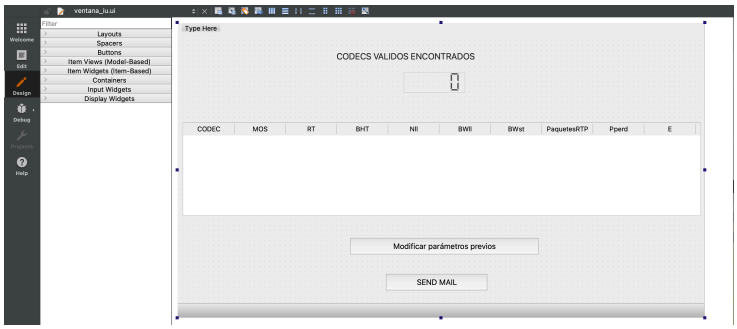

Figura 4: Interfaz de diseño QT creator

3.1.5. calculateCodec.py

Este archivo contiene una clase que incluye todo el cálculo teórico para adecuar los parámetros de entrada con el codec de audio más oportuno. Para ello hará uso de las siguientes clases y archivos:

- **erlangB.py** : dimensionado del número de lineas acorde al número de clientes y llamadas.
- **codecInfoDB.py**: información de todos los codecs y cabeceras de protocolos.

En este archivo se ha definido la clase **voipCodecs**. Esta contiene como datos miembro todos los valores necesarios para usar en la clase. Con todas estas funciones podemos obtener un resultado que nos permita elegir de forma objetiva cual es el CODEC de audio más adecuado para nuestra aplicación. Las funciones más importantes son las siguientes:

- getValidCodec: codecs validos dado un MOS.
- getBWst: Ancho de banda del enlace SIP-Trunk necesario.
- getNumberOfCalls: número total de llamadas que se podrán cursar a la vez.
- getBHT: tráfico máximo en hora cargada.
- getRetardoTotal: retardo total de boca-oreja.
- calculateAll: llama a todas las funciones para hacer todos los cálculos invocando únicamente a esta función.

3.1.6. main.py

Este es el archivo que debemos ejecutar con python3 para acceder a la interfaz que hemos implementado. Para ello hemos definido en este 2 clase:

- **MainWindow:** esta es la ventana de la interfaz donde se solicitan todos los parámetros de entrada y donde podemos ver elegir nuestros requisitos para que posteriormente acorde a estos se nos muestren en la pantalla de resultados. En esta clase se incluyen funciones que permiten navegar entre las distintas paginas. Podemos destacar las siguientes:
  - goToSecondPage: avanzar a la siguiente página donde aparecen más parámetros.
  - goToThirdPage: avanzar a la siguiente página donde aparecen más parámetros.
  - goToFourthPage: avanzar a la siguiente página donde aparecen más parámetros.
  - goToFifthPage: avanzar a la siguiente página donde aparecen más parámetros.
  - leerFichero: leer el fichero que contiene el vector de entrada de 0 y 1
  - calcular: hacer todos los calculos llamando a las funciones de la clase calculateCodec acorde a todos los parámetros de entrada.
- **ResultsWindow**: en esta clase se utiliza la ventana de resultados para mostrar todos los codecs validos que cumplen los requisitos de entrada que hemos fijado en la ventana MainWindow. Tiene 2 funciones principales:
  - recalculate: vuelve a la ventana principal en caso de que queramos modificar algún parámetro.
  - enviarCorreo: permite enviar un correo a la dirección deseada con los resultados obtenidos.

4. Ejecución del código

El archivo principal es main.py donde se definen y utilizan las clases de los demás archivos. La ejecución como detallaremos más adelante debe hacerse mediante el comando. Este comando dejará en backgroud corriendo en la terminal un proceso y abrirá una interfaz gráfica como la mostrada en figuras anteriores donde hemos de introducir los parámetros y posteriormente obtendremos un resultado.

El comando de ejecución debe lanzarse con la versión 3 de python de la siguiente forma:

```
python3 main.py
```

Más información sobre instalación y uso en: `https://github.com/joseluistejero/voipNetworkQoS` o siguiendo el video tutorial en `https://drive.google.com/file/d/1EhAPlbc8ZZEkpxPg19BYinM4ts9ylXab/view?usp=sharing`