Universidad del Istmo de Guatemala

Facultad de Ingeniería

Programming for Telecommunications

Ing. Mario Stuardo Porras

Documentación del proyecto final

Simulación de CAMEL

Guatemala, 10 de noviembre de 2022

Universidad del Istmo
Facultad de Ingeniería
Programming for Telecommunications
6to semestre

Integrantes

José Andrés Ramos Gutiérrez

Gabriel Eduardo Cuá Fagiani

Jorge Antonio Rosado Maldonado

Bill Rony J López Montúfar

Christopher Josué García Florián

Gustavo Adolfo Morales Xitamul

Luis David Serrano Conde

Universidad del Istmo
Facultad de Ingeniería
Programming for Telecommunications
6to semestre

Procedimiento del código fuente

Para dar inicio y que el proyecto se ejecutara de la mejor manera se importaron las siguientes librerías:

```
import socket
import sys
import pickle
import time
import random
from datetime import datetime
from typing import final
```

Luego de ello continuamos con definir la primera estructura de datos del proyecto, llamada IDP, función que tiene por objetivo enviar al servidor un calling party number, msc address y un called party number.

```
class IDP:
    callingPartyNumber =""
    mscAddress = ""
    calledPartyNumber = ""
```

Como segundo paso se procedió a definir la segunda clase de tipo mensaje, la cual en primera instancia contiene un RRB y un ERB.

```
class RRB:
    o_answer = bool
    o_disc = bool

class ERB:
    o_answer = bool
    o_disc = bool
```

La clase RRB dentro del proyecto tiene como función enviar un request del servidor al cliente para confirmar o analiza la información de la llamada.

Ahora bien, la clase ERB tiene como función enviar la información solicitada por la clase RRB.

```
s = socket.socket()
host = input(str("Please enter the mscAddress :
port = 8080
s.connect((host,port))
name = input(str("Please enter your username : "
print(" Connected to the central")
```

Después de haber definido cada una de las clases, continuamos con la creación del socket tanto en el servidor como en el cliente. Además de esto a través del comando s.connect se logró la vinculación entre host y puerto lo cual permitió el establecimiento de la conexión del socket.

```
s.send(name.encode())
s_name = s.recv(1024)
s_name = s_name.decode()
print("")
print(s_name, "has joined the chat room ")
tiempoacr = 0
```

Una vez establecida la conexión del socket, se inicia la transferencia de información entre servidor y cliente.

Dato: Lo primero que se recibe del lado del servidor será el nombre del cliente.

```
while 1:
    mensaje = IDP()
    mensaje.callingPartyNumber =  "3506013309"
    mensaje.mscAddress =  "91655340000512"
    mensaje.calledPartyNumber =  "3144582356"
    message = pickle.dumps(mensaje)
    s.send(message)
    print("IDP enviado")
    print("")
    data = s.recv(4096)
    data_variable = pickle.loads(data)
    print(s_name, ":" ,data_variable.continuar)
    print("")
```

La función del while es el recibir parámetros y definir cada uno de los mensajes enviados por la clase mensaje. Después de esto, se des-serealiza el flujo de datos, lo cual permite el imprimir los parámetros emoji y texto de la variable del mensaje enviado.

```
data = s.recv(4096)
data_variable = pickle.loads(data)                    #Des
print(s_name, ":" ,data_variable.o_answer,data_variable.o_disc)
print("RRB recibido")
print("")
time.sleep(random.randint(1,10))
erb = ERB()
erb.o_answer = True
erb.o_disc = False
erb = pickle.dumps(erb)
inicio = datetime.now()
s.send(erb)
print("ERB enviado (o_answer)", inicio)
print("")
```

Aquí sucede algo similar a lo anteriormente mencionado, se des-serealiza el flujo de datos para que la transmisión de datos pueda llegar a ser impresa, además de esto se incluye un time. Sleep el cual nos permite simular la función wait for instructions.

```
data = s.recv(4096)
data_variable = pickle.loads(data)
time.sleep(data_variable.tiempo)
tiempoacr = data_variable.tiempo
print("ACR(",tiempoacr,")")
print("")
data = s.recv(4096)
data_variable = pickle.loads(data)
time.sleep(data_variable.tiempo)
tiempoacr = tiempoacr + data_variable.tiempo
print("ACR(",tiempoacr,")")
print("")
data = s.recv(4096)
data_variable = pickle.loads(data)
time.sleep(data_variable.tiempo)
tiempoacr = tiempoacr + data_variable.tiempo
print("ACR(",tiempoacr,")")
print("")
```

En esta parte del código se realiza una sumatoria de tiempos del ACR, los cuales al final del código nos servirán para poder enviar un informe del proceso de la llamada, el cual incluye la duración total de la misma.

```
erb = ERB()
erb.o_answer = False
erb.o_disc = True
erb = pickle.dumps(erb)
finalizacion = datetime.now()
s.send(erb)
print("ERB enviado (o_disc) ", finalizacion)
print("Llamada finalizada")
print("Duracion:", finalizacion - inicio)
print("")

exit()
```

Por ultimo se declaro el proceso de finalización de llamada y él envió del reporte final, el cual incluye la duración de la llamada, cuando y cuando termino.

Universidad del Istmo
Facultad de Ingeniería
Programming for Telecommunications
6to semestre

Diagrama