iQoS

Pruebas de calidad de servicio en VoIP con Python y Linux

Pycon Medellín 2018

ACERCA DE MI ...

Oscar Maestre Sanmiguel

Ingeniero de sistemas e informática

Estudiante de Máster en Data Science - UOC

Amante de tecnologías Open Source, Python y Linux

oscarlibre@gmail.com

¿QUÉ ES IQOS?

- Software Open Source escrito en Python
- Integra múltiples herramientas de GNU/Linux
- Realiza pruebas de QoS en tiempo real
- Emula llamadas telefónicas de VoIP
- Evalúa métricas de QoS

¿PROBLEMA?

- VoIP es sensible a las distorsiones en la voz, retardos y fluctuaciones
- Comparte el tráfico con otros protocolos y servicios
- La calidad del servicio depende de ciertos parámetros que deben ser evaluados en la infraestructura de red y en el canal de comunicación.
- Software especializado para realizar mediciones costoso y en muchas ocasiones complicados de usar.

VENTAJAS

- Viabilidad para implementar VoIP
- Integración de diversas herramientas de redes
- Apoyo en el análisis de métricas
- Herramienta de software libre
- Interfaz Web de fácil uso

CONCEPTOS

CALIDAD DE SERVICIO EN VOIP

La calidad de servicio se define como un conjunto de tecnologías y técnicas aplicadas a las redes de datos, cuyo objetivo es intentar garantizar el buen desempeño de la red [1].

ESTÁNDARES

RTP

Desarrollado por la IETF.

RFC 3550 [2].

Transmisión y recepción de contenido de audio y video en tiempo real.

Facilita las mediciones de métricas de QoS.

Permite la creación de perfiles y la extensión de las cabeceras en los paquetes

CODEC G.711

Recomendación ITU-T G.711

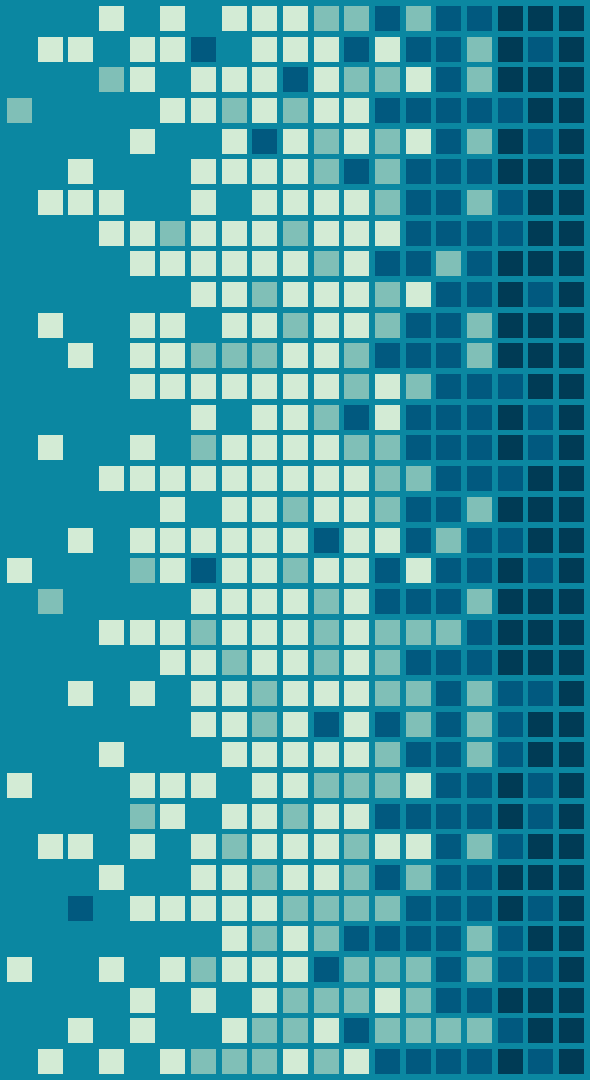
Frecuencia de muestreo: 8000 Hz

Número de bits: 8

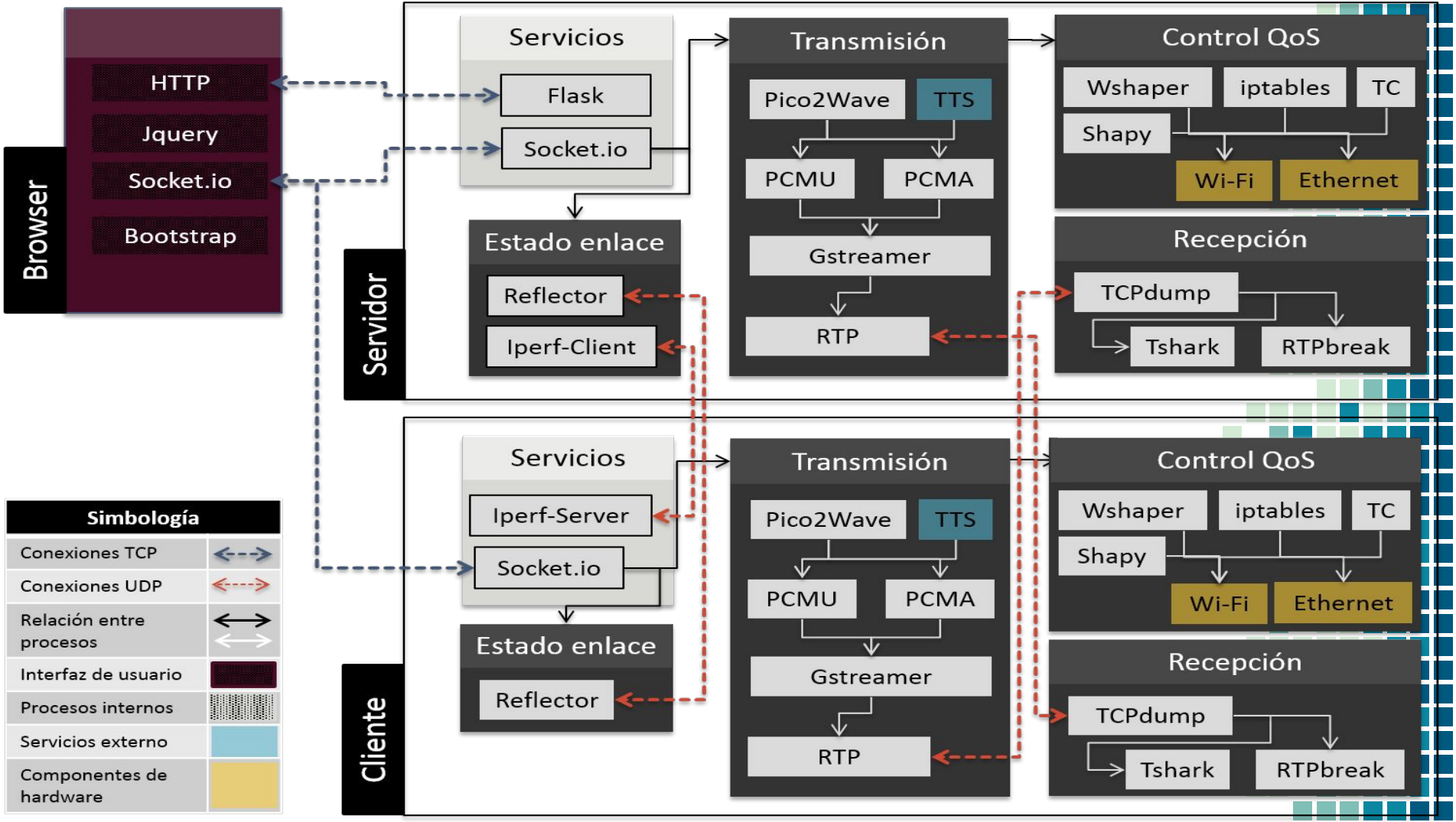
Usado en la red de telefonía conmutada

La utilización de un códec determinado va a tener repercusiones que afectan directamente el ancho de banda y la calidad de la voz

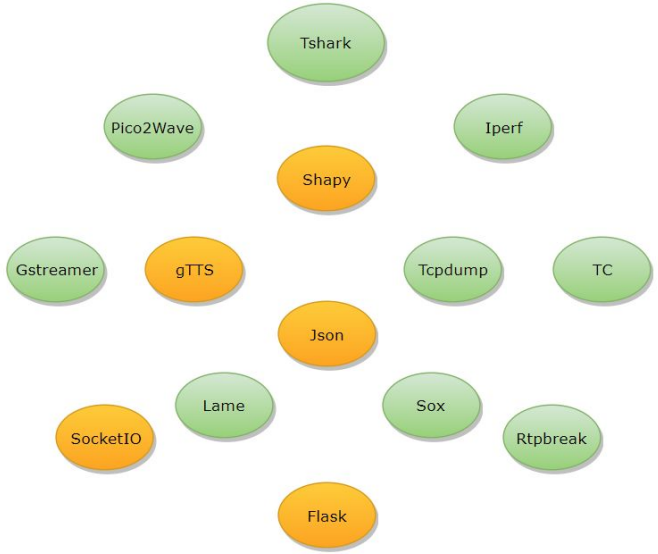
“La utilización de un códec determinado va a tener repercusiones que afectan directamente el ancho de banda y la calidad de la voz



ARQUITECTURA


Browser
HTTP
Jquery
Socket.io
Bootstrap
Servidor
Servicios
Flask
Socket.io
Estado enlace
Reflector
Iperf-Client
Transmisión
Pico2Wave
TTS
PCMU
PCMA
Gstreamer
RTP
Control QoS
Wshaper
iptables
TC
Shapy
Wi-Fi
Ethernet
Recepción
TCPdump
Tshark
RTPbreak
Cliente
Servicios
Iperf-Server
Socket.io
Estado enlace
Reflector
Transmisión
Pico2Wave
TTS
PCMU
PCMA
Gstreamer
RTP
Control QoS
Wshaper
iptables
TC
Shapy
Wi-Fi
Ethernet
Recepción
TCPdump
Tshark
RTPbreak
Simbología
Conexiones TCP
Conexiones UDP
Relación entre procesos
Interfaz de usuario
Procesos internos
Servicios externo
Componentes de hardware

MÓDULOS Y HERRAMIENTAS



Tshark	-	Extraer métricas
Shapy[3]	-	Emulacion de métricas
Iperf	-	Medición de BW
TC	-	Emular perdida paquetes
tcpdump	-	Captura de paquetes
Rtpbreak	-	Detectar sesiones RTP
Flask	-	Microframework Web
SocketIO	-	Websockets en Python
Gstreamer	-	Transmision de RTP
Pico2wave	-	TTS Linux
gTTS	-	Google TTS
Sox	-	Conversión de audios
LAME	-	Codificador mp3
Json	-	Intercambio de datos

SHAPY

```
from shapy.emulation.shaper import Shaper

ps = {("127.0.0.2",) : {'upload': 1024, 'download': 1024, 'delay': 56},
      ("127.0.0.3",) : {'upload': 256, 'download': 512, 'delay': 30},
      ("127.0.0.4",) : {'upload': 256, 'download': 512, 'delay': 30},
}

sh = Shaper()
sh.set_shaping(ps)
```

gTTS

1. Import `gTTS`

```
>> from gtts import gTTS
```

2. Create an instance

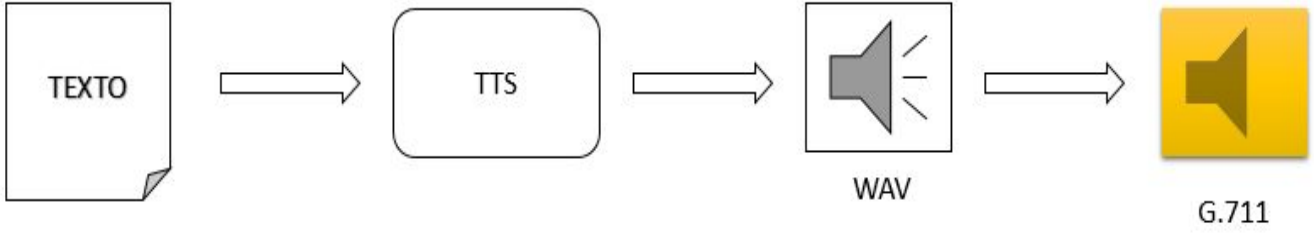
```
>> tts = gTTS(text='Hello', lang='en', slow=True)
```

3. Write to a file

- *To disk* using `save(file_name)`

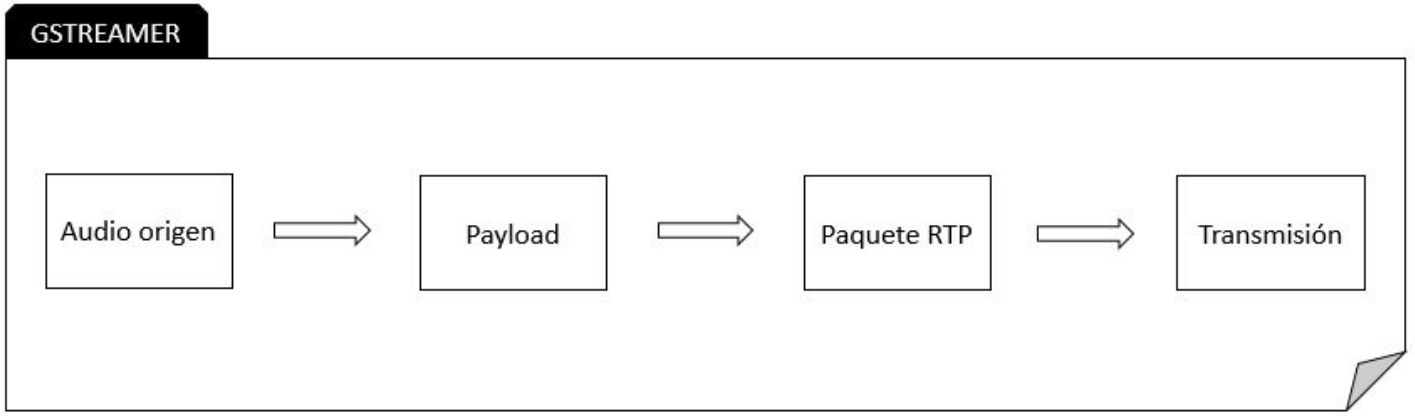
```
>> tts.save("hello.mp3")
```

GENERACIÓN DE AUDIO



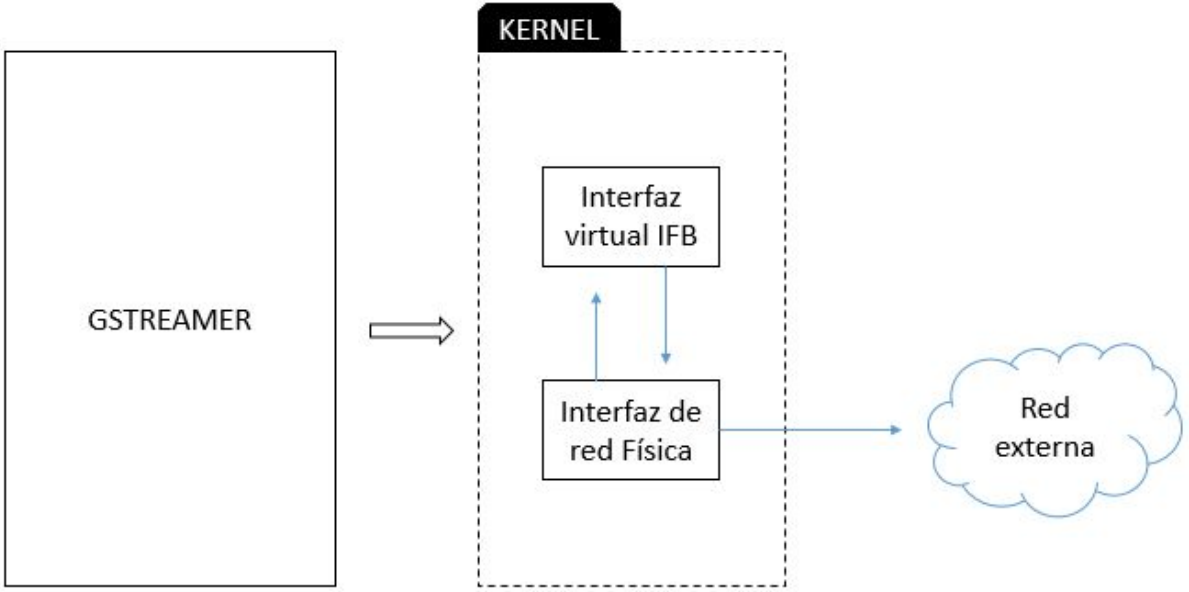
```
if server:
    texto1  =   "Servidor, llamada "    +   str(indice)
    audio   =   "audio_hombre.mp3"
    idioma  =   "es-es"
else:
    texto1  =   "Cliente, llamada " +   str(indice)
    audio   =   "audio_mujer.mp3"
    idioma  =   "es-us"
texto2  =   "Fin de llamada "   +   str(indice)
if not local:
    pico2wave1  =   os.system("pico2wave -l es-ES -w " + ruta + "inicio.wav" + " '" + texto1    + "'")
    pico2wave2  =   os.system("pico2wave -l es-ES -w " + ruta + "final.wav" + " '" + texto2 + "'")
    audio="gtts.mp3"
    tts         =   gTTS(text=texto, lang=idioma)
    tts.save(ruta + audio)
    mp3_wav     =   os.system("sox " + ruta + audio + " -r 16k " + ruta + audio.replace('.mp3','.wav'))
    combinar    =   os.system("sox " + ruta + "inicio.wav " + ruta + audio.replace('.mp3','.wav') +
                " " + ruta + "final.wav " + ruta+archivo.replace('.mp3','.wav'))
    wav_mp3     =   os.system("lame "+ ruta + archivo.replace('.mp3','.wav') +  " " + ruta+archivo)
    return True
else:
    #CREA MARCAS WAV CLIENTE/SERVIDOR LLAMADA N
    pico2wave1  =   os.system("pico2wave -l " + cod_lenguaje + " -w " + ruta + "inicio.wav" + " '" + texto1 + "'")
    pico2wave2  =   os.system("pico2wave -l " + cod_lenguaje + " -w " + ruta + "final.wav" + " '" + texto2  + "'")
    copia       =   os.system("cp static/pruebas/default/audios/" + audio + " " + ruta)
    mp3_wav     =   os.system("sox " + ruta + audio + " -r 16k " + ruta+audio.replace('.mp3','.wav'))
    combinar    =   os.system("sox " + ruta + "inicio.wav " + ruta+audio.replace('.mp3','.wav') + " " + ruta +
                    "final.wav " + ruta+archivo.replace('.mp3','.wav'))
    wav_mp3     =   os.system("lame "+ ruta + archivo.replace('.mp3','.wav') +  " " + ruta+archivo)
    return True
```

TRANSMISIÓN



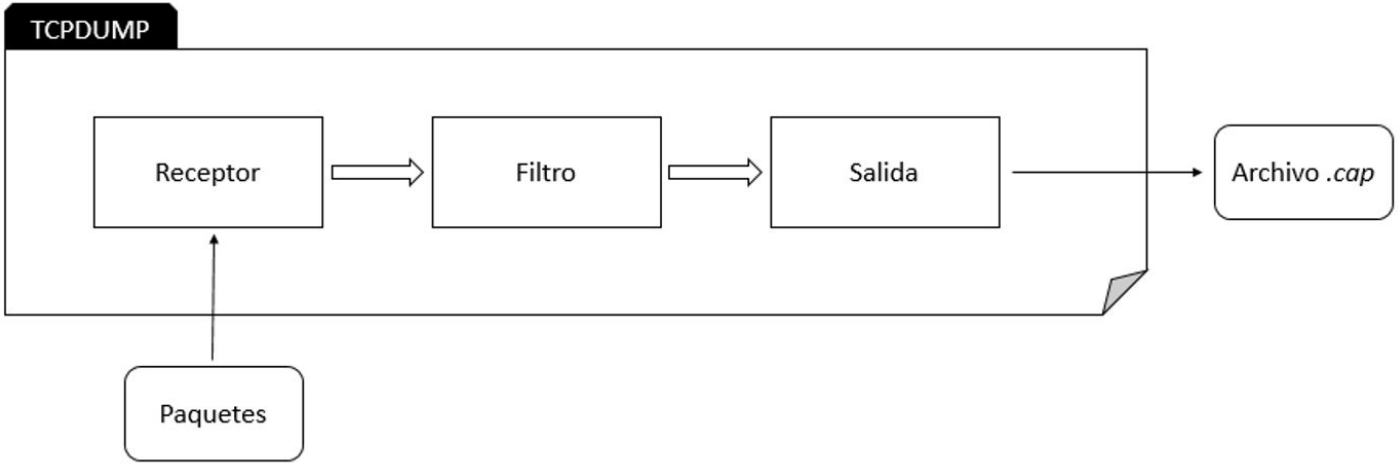
```
#TRANSMITE PAQUETES RTP EN LA INTERFAZ SELECCIONADA
def txAudio(self, audio_generado, codec, encolar = False):
    try:
        self.auout  =   audio_generado
        comando     =   "gst-launch-0.10 -v "
        archivo     =   "filesrc location=" + self.auout + " "
        marca       =   "do-timestamp=true ! "
        #pcm            =   "audio/x-alaw, rate=8000, channels=1  ! "
        #rtp            =   "rtppcmapay ! "
        pcm     =   "mad ! audioconvert ! audioresample ! alawenc ! "
        rtp     =   "rtppcmapay !"
        destino     =   "udpsink host=" + self.ipd + " port=" + str(self.pd)

        if codec == "pcmu":
            pcm     =   "mad ! audioconvert ! audioresample ! mulawenc ! "
            rtp     =   "rtppcmupay ! "

        if  encolar:
            print "entra"
            rtp     =   "queue ! " + rtp

        #ASEGURA EL MULTIHILO CON COMANDOS DEL SISTEMA
        salida, error = Popen(comando+archivo+marca+pcm+rtp+destino, stdin=PIPE,
                        stdout=PIPE, stderr=PIPE, close_fds=True, shell=True).communicate(None)
```

CONTROL


KERNEL
GSTREAMER
Interfaz virtual IFB
Interfaz de red Física
Red externa

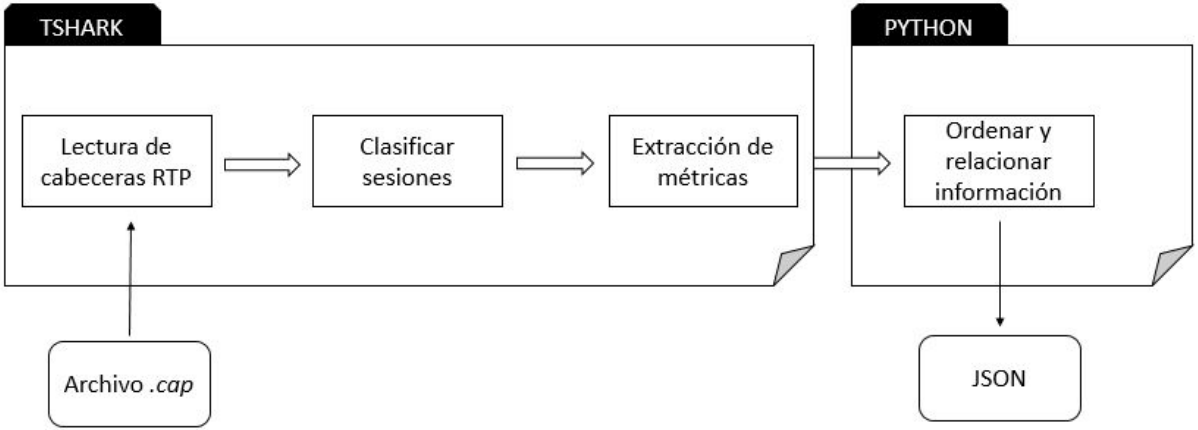
```
#APLICA EL TRAFIC SHAPING PARA EMULAR LATECIA, PERDIDAS, JITTER
def controlar(self):
    try:
        from shapy.emulation.shaper import Shaper
        from shapy import register_settings
        register_settings('configuracion')
        regla = {(self.ipo,) : {'upload': self.bw_up, 'download': self.bw_down,
                'delay': self.delay, 'jitter': self.jitter},}
        sh          =   Shaper()
        sh.set_shaping(regla)
```

RECEPCIÓN



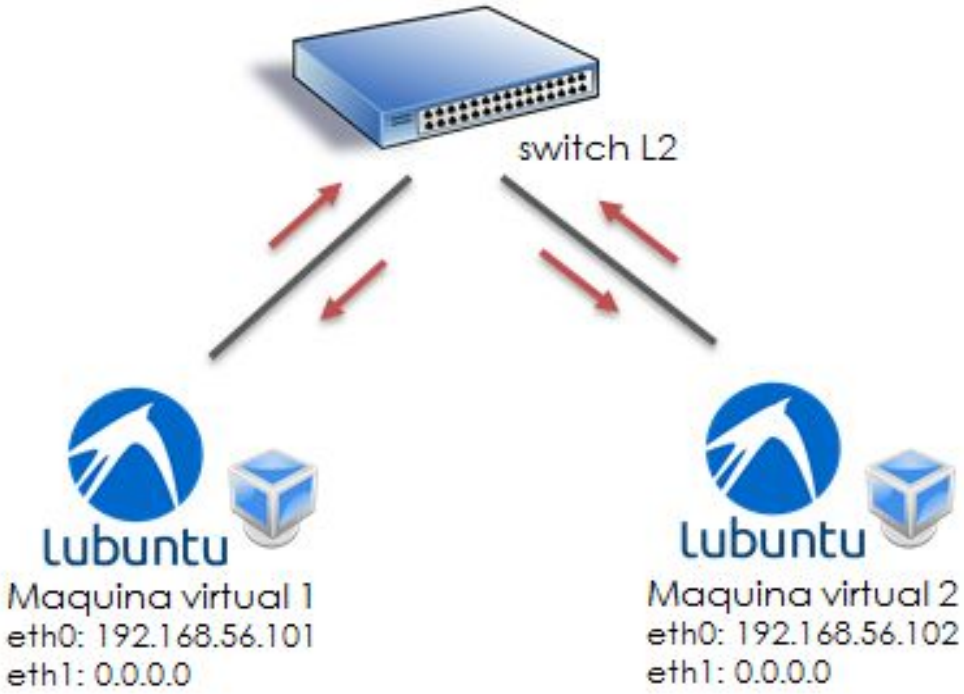
```
#CAPTURA PAQUETES EN LA INTERFAZ SELECCIONADA
def rxAudio(self, nombre_prueba,codec):
    try:
        comando     =   "sudo tcpdump -i "
        interfaz    =   self.iface + " "
        puerto      =   self.pd
        filtro      =   "'(src " + self.ipd + " and dst " + self.ipo + ") or (src " +
                        self.ipo + " and dst " + self.ipd +") and udp' -w "
        archivo_cap =   nombre_prueba
        filtro2     =   '\'(src ' + self.ipd + ' and dst ' + self.ipo + ') or (src ' +
                        self.ipo + ' and dst ' + self.ipd +') and udp\' -w '
        self.pcap_recv  =   archivo_cap
        self.codec      =codec
        verbose     =   " -vv -ttttt"
        #POPEN PERMITE CREAR HILOS USADO EN COMANDOS EN ESCUCHA QUE NO FINALIZAN
        salida1, error =    Popen(comando + interfaz + filtro2 + archivo_cap + verbose, stdin=PIPE,
                            stdout=PIPE, stderr=PIPE, close_fds=True, shell=True).communicate(None)
```

PROCESAMIENTO



```
#ANALIZA EL TRAFICO CAPTURADO, EXTRAE METRICAS RTP
def analizar(self):
    try:
        comando =   "tshark -r "
        filtro  =   " -o    rtp.heuristic_rtp:TRUE -nqz rtp,streams"
        salida = commands.getoutput(comando + self.pcap_recv + filtro).split('\n')
        i=0
        print "TSHARK", salida
```

DEMO


switch L2
Lubuntu
Maquina virtual 1
eth0: 192.168.56.101
eth1: 0.0.0.0
Lubuntu
Maquina virtual 2
eth0: 192.168.56.102
eth1: 0.0.0.0

iQos

Configuración

Dirección IP remota	192.168.56.101
Interfaz de red servidor	eth1
Interfaz de red cliente	eth1
Número de llamadas	1
Latencia (ms)	0
Paquetes perdidos (%)	0
Jitter (ms)	0
Upstream (Kbps)	
Downstream (Kbps)	
Codec	PCMA

Texto de prueba

Iniciar | Evaluación preliminar

Llamada:

Evaluación Preliminar

Test de velocidad

Resultado

Upstream:	98831.04 Kbits/sec
Downstream:	99024.84 Kbits/sec
Latencia:	1.41 ms
Jitter:	0.00 ms
Pérdida de paquetes:	0.0%

iQos

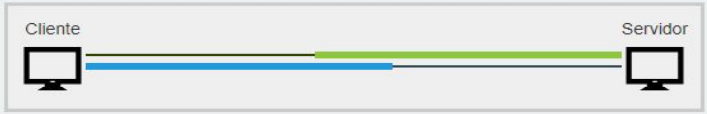
<

Llamada:

Configuración

Dirección IP remota	192.168.56.101
Interfaz de red servidor	eth1
Interfaz de red cliente	eth1
Número de llamadas	15
Latencia (ms)	100
Paquetes perdidos (%)	5
Jitter (ms)	50
Upstream (Kbps)	1000
Downstream (Kbps)	1000
Codec	PCMA

Texto de prueba



Iniciar

Evaluación preliminar

[Servidor]> Llamando

Llamada: General

Cliente				Sentido & Lantencia (ms)	Servidor			
Jitter		Paquetes			Paquetes		Jitter	
Máx (ms)	AVG (ms)	✓	X		✓	X	Máx (ms)	AVG (ms)
20.90	9.64	2753	132	> 223.66	2743	131	20.90	9.63
28.07	9.11	2700	143	< 232.02	2698	142	28.14	9.12

Audios

Audio generado

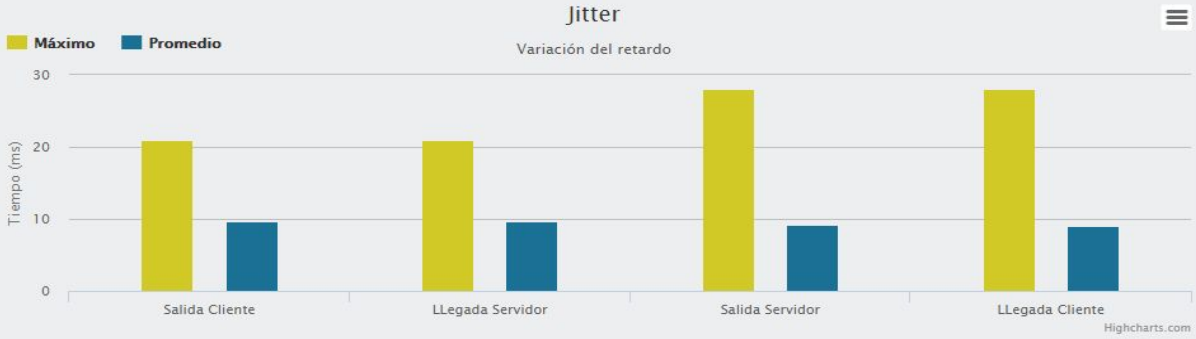
Audio codificado

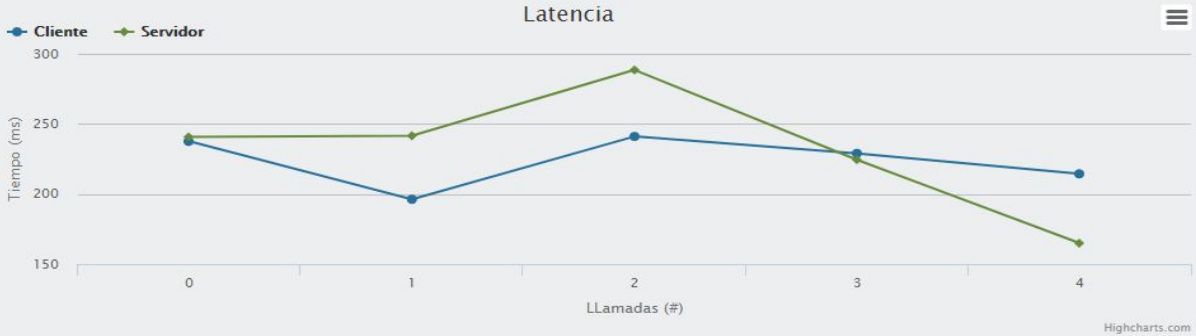
Descargar captura Cliente

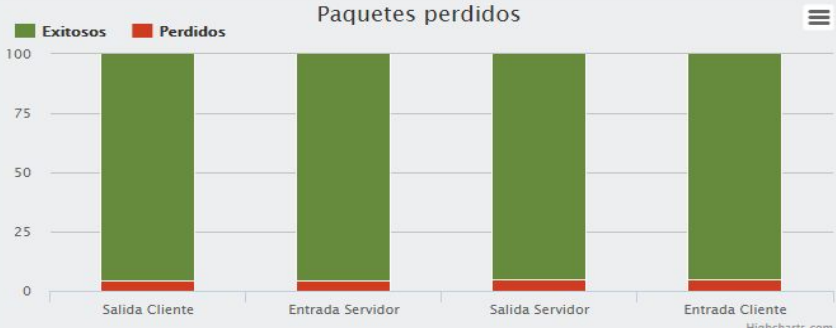
Descargar captura Servidor

Audio recibido

Reproductor



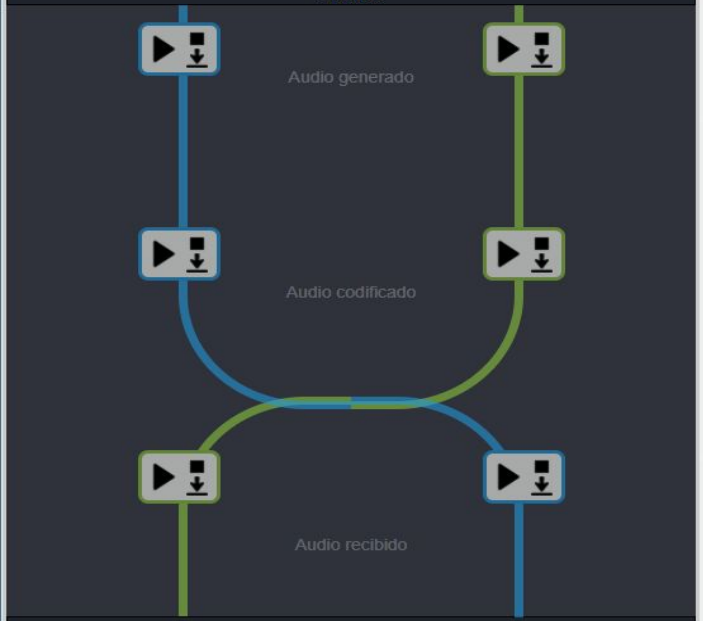




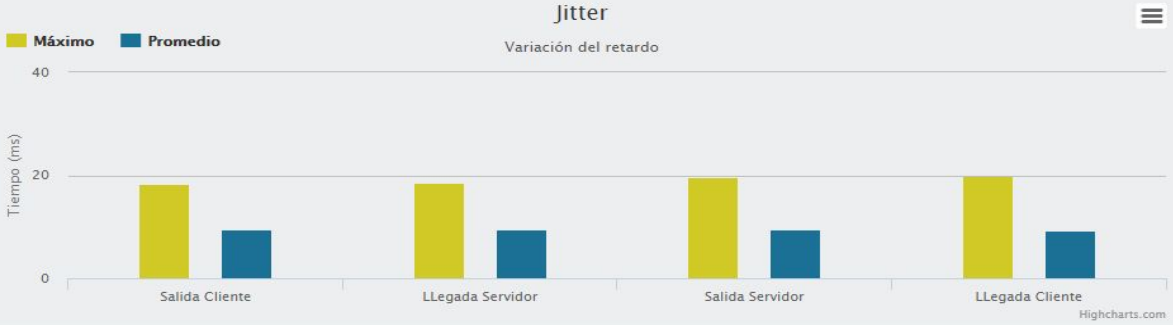
Llamada: 0

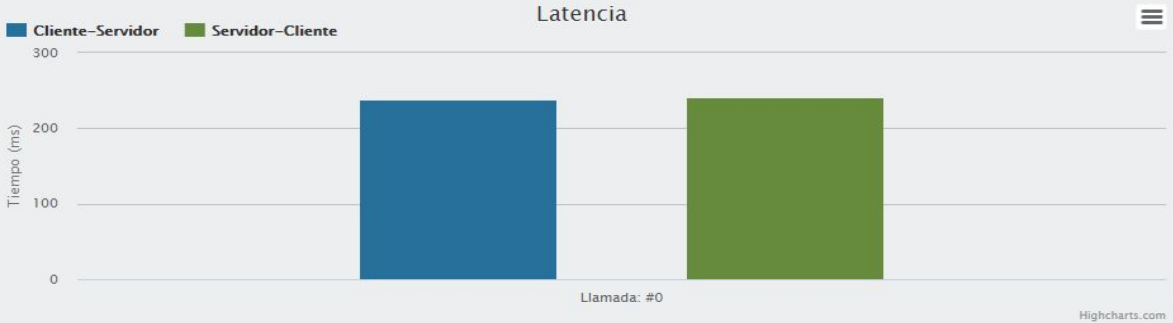
Cliente				Sentido & Lantencia (ms)	Servidor			
Jitter		Paquetes			Paquetes		Jitter	
Máx (ms)	AVG (ms)	✓	X		✓	X	Máx (ms)	AVG (ms)
18.41	9.52	547	30	> 237.60	546	30	18.50	9.50
19.92	9.39	541	28	< 240.71	541	28	19.68	9.40

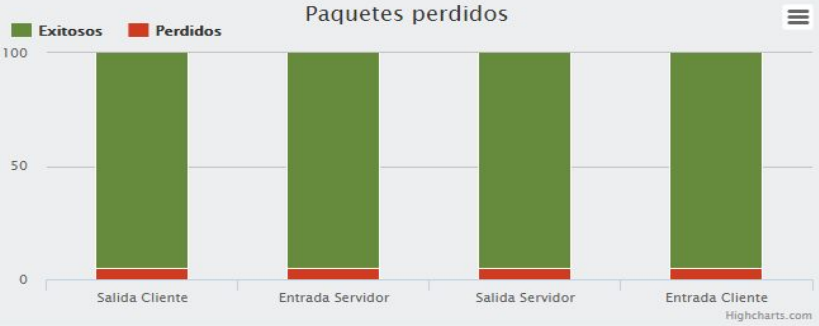
Audios



Reproductor







GRACIAS !

Any questions?

REFERENCIAS

[1]Unión Internacional de las Telecomunicaciones. Recomendación Y.1540 En línea. Disponible en: https://www.itu.int/rec/dologin_pub.asp?lang=s&id=T-REC-Y.1540-201103-I!!PDF-E&type=items

[2] RFC 3550. RTP: A Transport Protocol for Real-Time Applications. Julio de 2005

[3] Shapy . Disponible en: https://github.com/praus/shapy