

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

“Adición de texto en tiempo real a un video utilizando una arquitectura IMS”

Proyecto Tecnológico

Alumno:

Juan Manuel Cruz Pérez

208202284

Asesor:

M. en C. Arturo Zúñiga López

Mexico D.F.

Enero 2015

## DECLARATORIA

Yo, Arturo Zúñiga López, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



---

Arturo Zúñiga López

Yo, Juan Manuel Cruz Pérez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



---

Juan Manuel Cruz Pérez

# Índice

<b>Introducción.....</b>	<b>5</b>
<b>Objetivos.....</b>	<b>7</b>
<b>Antecedentes .....</b>	<b>8</b>
<b>IMS (IP Multimedia Subsystem).....</b>	<b>8</b>
<b>ASTERISK .....</b>	<b>9</b>
<b>SIP .....</b>	<b>10</b>
<b>OPENCV .....</b>	<b>11</b>
<b>SOCKETS .....</b>	<b>11</b>
<b>Desarrollo .....</b>	<b>13</b>
<b>Instalación y configuración de Asterisk.....</b>	<b>13</b>
<b>Clientes y configuración .....</b>	<b>15</b>
<b>Configuración de los sofphone (Zoiper) .....</b>	<b>18</b>
<b>Instalación de SIPDUMP .....</b>	<b>20</b>
<b>Código Script bash.....</b>	<b>22</b>
<b>Explicación del código .....</b>	<b>23</b>
<b>Ejemplo del proyecto .....</b>	<b>31</b>
<b>Resultados .....</b>	<b>34</b>
<b>Conclusión.....</b>	<b>37</b>
<b>Bibliografía.....</b>	<b>38</b>
<b>Apéndice A.....</b>	<b>39</b>

## Resumen

Este proyecto como su nombre lo indica consta de la adición de texto a video en tiempo real utilizando una arquitectura IMS<sup>1</sup>, a grandes rasgos consiste en lo siguiente, se toman 2 sofphone, uno realiza la llamada a otro, mientras en una terminal se está reproduciendo video, tiene que aparecer en el video que está en reproducción, la persona que está llamando.

Para la elaboración de este proyecto y su correcto funcionamiento, se utilizaron las tecnologías y herramientas, que se mencionan a continuación con una breve descripción: Empezamos con *Asterisk*, esta herramienta nos permitió registrar los sofphone y llevar un control de las llamadas que realizamos desde nuestros sofphones<sup>2</sup>, precisamente para nuestros sofphone utilizamos el software llamado Zoiper, teniendo lista nuestra central telefónica con sus respectivas extensiones funcionando, ocupamos SIPdump para capturar los paquetes SIP<sup>3</sup>, se para el *streaming* de video, se probaron varias tecnologías como, ffmpeg, nos dimos cuenta que este software no añade texto en tiempo real, después se intento con Html5 y JavaScript, se nos dificulto un poco ya que no conseguimos visualizar el flujo de video, por ultimo intentamos programando sockets y lo conseguimos, con ayuda de la librería Opencv, esta librería cuenta con funciones que nos permitió terminar el proyecto con éxito.

Se concluye que es posible añadir texto en tiempo real sobre un flujo de video, con las herramientas ya mencionadas, ¿Por que con las herramientas ya mencionadas?, porque se realizaron pruebas con otras y no alcanzamos el mismo resultado, por ejemplo en primera instancia se pretendía usar Wireshar para capturar el tráfico SIP, y este software lo hace, pero a la hora de filtrar los paquetes nos causo muchos problemas, así que decidimos utilizar SIPdump, sin embargo aunque se cumplió con el objetivo principal, tuvimos detalles como, el texto que se despliega sobre el flujo de video, aparece al revés, así como pequeños retardos en el video debido al hardware utilizado.

---

<sup>1</sup> IMS: Subsistema Multimedia IP o IP Multimedia Red Central Subsystem

<sup>2</sup> Sofphone: software que es utilizado para realizar llamadas a otros softphones o a otros teléfonos

<sup>3</sup> Session Initiation Protocol: es un protocolo de control y señalización

## Introducción

Se puede pensar en la sociedad actual como la sociedad de las comunicaciones, donde todos y cada uno de nosotros ha tenido que adoptar nuevas y cada vez más sofisticadas tecnologías que nos han llevado a un mundo donde el estar comunicados es requisito indispensable para vivir en sociedad. En la actualidad no basta solo con poder comunicarnos de manera tradicional, si no que la aparición de nuevos y revolucionarios servicios han creado un entorno en donde la interactividad y la comunicación no solo por voz sino por video se está convirtiendo en la manera tradicional de comunicarnos.

El internet ha permitido desde hace varios años, el uso de numerosos servicios exitosos tales como el correo electrónico o “E-mail”, el “Web”, el “streaming” de audio y video, los cambios interactivos o “chat”, con una calidad muy aceptable.

Con la demanda que hay en las telecomunicaciones, juega un papel muy importante, el IMS o “IP Multimedia Subsystem” para el mundo de las telecomunicaciones es una nueva arquitectura basada en nuevos conceptos, nuevas tecnologías, la arquitectura soporta sobre una red IP, las sesiones aplicativos en tiempo real (voz, video, conferencia). IMS trabaja utilizando el protocolo SIP, es un protocolo de señalización que permite el establecimiento, la liberación y la modificación de sesiones multimedia.

El IMS en las redes fijas y móviles representa un cambio fundamental, es concebido para ofrecer a los usuarios la posibilidad de establecer sesiones multimedia usando todo tipo de acceso de alta velocidad y una conmutación de paquetes IP.

SIP es usado en el IMS como protocolo de señalización para el control de sesiones y el control de servicio. Las solicitudes SIP son satisfechas por respuestas identificadas por un código numérico.

Una terminal IMS se trata de una aplicación sobre un equipo de usuario que emite y recibe solicitudes SIP.

En la actualidad se está gestando una increíble revolución en el campo de las tecnologías. Particularmente, en la industria de las telecomunicaciones, la revolución empezó con la centralita PBX<sup>4</sup> *open source*<sup>5</sup> llamada Asterisk y los sistemas basados sobre VoIP (Voz sobre IP). Hoy en día existen muchas empresas del sector de las telecomunicaciones que ofrecen servicios y soluciones telefónicos. El inconveniente de estas soluciones es que son cerradas, propietarias y la gran mayoría costosas. Por el contrario la centralita telefónica Asterisk proporciona un estándar de comunicaciones VoIP, lo que permite no estar sujeto a las limitaciones de ningún fabricante.

---

<sup>4</sup> PBX: siglas en inglés de Private Branch Exchange (Central Secundaria Privada Automática o central telefónica).

<sup>5</sup> Open source: código abierto.

Asterisk es un programa de software libre que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de VoIP. Lo más interesante de Asterisk es que reconoce muchos protocolos VoIP como pueden ser SIP, H.323, IAX. Asterisk puede interoperar con terminales IP actuando como un registrador y como *Gateway* entre ambos.

Otra de las herramientas que forman parte fundamental de este proyecto es el Sipdump<sup>6</sup>, esta herramienta de captura de paquetes SIP, escucha una interfaz especificada para llamadas SIP.

Finalmente los sockets que se utilizaron como un último recurso para la realización de este proyecto, pero ¿Qué son los sockets? Un *socket*(enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (*API, application programming interface*).

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como *FTP*<sup>7</sup>, *Gopher*<sup>8</sup>, o *WWW*).

---

<sup>6</sup> SIP: es una herramienta de captura de paquetes SIP.

<sup>7</sup> FTP: 'Protocolo de Transferencia de Archivos'.

<sup>8</sup> Gopher: es un servicio de Internet consistente en el acceso a la información a través de menús.

## Objetivos

### Objetivo general.

Realizar un prototipo de una red con arquitectura IMS, sobre la cual se comunicaran dos softphone, y en otra terminal que se está reproduciendo video se desplegara en tiempo real sobre tal video la información de quien está realizando la llamada con la leyenda “te está llamando Juan”.

### Objetivos específicos.

- Implementar una red con arquitectura IMS (*IP Multimedia Subsystem*).
- instalar un Proxy SIP.
- Instalar un servidor de video.
- Instalar en un par de computadoras un softphone.
- Analizar y capturar paquetes en una red.
- Detectar y capturar ciertos *streamings*.
- Programar un modulo que seleccione la información de los paquetes capturados.
- Programar un modulo que adicione o añada el texto en el *streaming*<sup>9</sup> de video.
- Direccionar el texto para que se despliegue sobre el video que se está ejecutando.

---

<sup>9</sup> Streaming: difusión en flujo.

## Antecedentes

Las tecnologías que se utilizan en telecomunicaciones han ido ingresando en distintos momentos en el tiempo, y por lo tanto siendo soportadas por plataformas diversas específicas para cada una de ellas, principalmente en concordancia con la tecnología que existía en los surgimientos de cada uno de los medios de comunicación.

Inicialmente las arquitecturas tecnológicas que soportaban cada una de las redes de comunicación estaban separadas, y utilizaban protocolos distintos. La televisión usa altas frecuencias y ultra altas frecuencias, los teléfonos móviles usan GSM<sup>10</sup> y las computadoras personales internet (a través del protocolo TCP/IP en la mayoría de los casos). Sin embargo los equipos de acceso a estas redes incorporaban cada vez más tecnología que permite obtener diversos contenidos multimedia. Los teléfonos móviles comenzaron a incorporar imágenes, música y video. Además existía un desafío en poder mantener las sesiones de los teléfonos móviles cuando cambiaban de red, ya sea a otra del mismo proveedor u otro diferente a través de *Roaming*<sup>11</sup>.

## IMS (IP Multimedia Subsystem)

El grupo *3rd Generation Partnership Project* se planteó crear el IMS que pretende ser una arquitectura que soporte el tráfico de voz, datos y multimedia mediante la conmutación de paquetes a direcciones IP, y con independencia del medio de acceso: teléfonos móviles, fijos; computadoras personales; y todo dispositivo que pueda tener una dirección IP en la red. Sólo requiere que los equipos utilicen el protocolo de sesión SIP (*Session Initiation Protocol*) que permite la señalización y administración de sesiones.

IMS define tres tipos de servidores de aplicación: Servidores SIP, OSA y CAMEL. Los SIP se comunican directamente con los S-CSCF<sup>12</sup> a través del protocolo SIP. Los servidores OSA cumplen la misma función, pero requieren el uso de un servidor SCS ('Service Capability Server') como intermediario entre el servidor OSA y el S-CSCF para traducir mensajes SIP. El servidor CAMEL, es un conjunto de mecanismos que permiten al operador de la red entregar servicios específicos de operador a los usuarios, a través de IP-SSF, que traduce las solicitudes CAMEL a solicitudes SIP.

Un AS puede contener más de una aplicación IMS. De esta forma, AS utiliza e interpreta los mensajes SIP enviados por el S-CSCF para enviar de vuelta una respuesta a través de este mismo servidor.

---

<sup>10</sup> GSM: sistema global para las comunicaciones móviles

<sup>11</sup> Roaming: se refiere a la capacidad de cambiar de un área de cobertura a otra sin interrupción en el servicio o pérdida en conectividad.

<sup>12</sup> S-CSCF: es un proxy SIP que es el primer punto de contacto para el terminal IMS



IMS fue diseñado para dar soporte amplio y complejo a los servicios multimedia IP para un alto número de usuarios. A la vez, los servidores CSCF pueden ser asignados dinámicamente a los usuarios, permitiendo escalabilidad independiente del nivel de tráfico. Los servidores son distribuidos de tal modo que la capacidad es extensible. Además el protocolo usado es SIP, que al ser basado en texto es fácil de depurar, pero el tamaño de los mensajes es grande. La arquitectura IMS, basada en capas (Acceso, Control y Servicio) y el uso de interfaces abiertos, hace que la implementación de un nuevo servicio o capacidad sea más fácil que en otro tipo de arquitecturas de red, reduciéndose el *Time to Market*<sup>13</sup>. Por esta misma razón, este tipo de redes son más escalables y flexibles, integrando soluciones, servidores o aplicaciones a terceros.

Antes de que la tecnología *streaming* apareciera en abril de 1995, la reproducción de contenido Multimedia a través de internet necesariamente implicaba tener que descargar completamente el "archivo contenedor" al disco duro local. Como los archivos de audio — y especialmente los de vídeo— tienden a ser enormes, su descarga y acceso como paquetes completos se vuelven una operación muy lenta.

Desde la década de los noventa, Internet ha revolucionado las industrias culturales, “desestabilizando la dicotomía entre los medios de masas y la comunicación interpersonal” Se puede decir que Internet se ha convertido en un medio que nos proporciona experiencias dentro de nuestra vida cotidiana, tanto de consumo, como de ocio y que ha entrado en una fase de desarrollo y de madurez a lo largo de los últimos años con la ayuda de la tecnología.

Internet en general y el consumo de Televisión y entretenimiento a través de la red en concreto, han supuesto un paso decisivo en la historia de la humanidad donde el complejo escenario lo constituyen nuevos usuarios que día a día ingresan a la web con la finalidad de encontrar cosas nuevas y entretenerse.

## ASTERISK

Asterisk fue creada en 1999 por Mark Spencer de la empresa Digium y donada a la comunidad con licencia libre tras lo cual se han recibido muchas colaboraciones y mejoras por parte de muchos desarrolladores libres y empresas sin solicitar nada a cambio.

Poco a poco, esta aplicación se ha convertido en la evolución de las tradicionales centralitas analógicas y digitales permitiendo también integración con la tecnología más actual: VoIP. Asterisk se convierte así en el mejor, más completo, avanzado y económico sistema de comunicaciones existente en la actualidad.

---

<sup>13</sup> Time to Market: es la longitud de tiempo que toma a partir de que un producto que se está concebido hasta que esta disponible para la venta.

Otro aliciente es su capacidad de ser programada, permitiendo realizar labores que hasta el día de hoy lo llevaban realizando sistemas extremadamente costosos y complicados y, gracias a Asterisk, esta misma labor se realiza de una forma más económica lo que fomenta el uso de sistemas libres como Linux y estándares abiertos como SIP, H323 o IAX.

Una de las ventajas más interesantes es su posibilidad como sistema híbrido, ya que permite gestionar comunicaciones telefónicas tradicionales (analógicas, digitales, móviles, ...) como comunicaciones IP mediante el uso de los protocolos estandar de VoIP.

## SIP

SIP, o *Session Initiation Protocol* es un protocolo de control y señalización usado mayoritariamente en los sistemas de Telefonía IP, que fue desarrollado por el IETF<sup>14</sup> (RFC 3261). Dicho protocolo permite crear, modificar y finalizar sesiones multimedia con uno o más participantes y sus mayores ventajas recaen en su simplicidad y consistencia.

Hasta la fecha, existían múltiples protocolos de señalización tales como el H.323 de la ITU, el SCCP de Cisco, o el MGCP, pero parece que poco a poco SIP está ganando la batalla del estándar: Cisco está progresivamente adoptando SIP como protocolo en sus sistemas de telefonía IP en detrimento de H.323 y SCCP, Microsoft ha elegido SIP como protocolo para su nuevo OCS (Office Communication Server), y los operadores (de móvil y fijo) también están implantando SIP dentro de su estrategia de convergencia, aprovechando de este modo la escalabilidad y interoperabilidad que nos proporciona el protocolo SIP.

## Funciones SIP

El protocolo SIP actúa de forma transparente, permitiendo el mapeo de nombres y la redirección de servicios ofreciendo así la implementación de la IN (*Intelligent Network*) de la PSTN o RTC.

Para conseguir los servicios de la IN el protocolo SIP dispone de distintas funciones. A continuación se enumeran las más importantes:

- Localización de usuarios (SIP proporciona soporte para la movilidad).
- Capacidades de usuario (SIP permite la negociación de parámetros).
- Disponibilidad del usuario
- Establecimiento y mantenimiento de una sesión.

En definitiva, el protocolo SIP permite la interacción entre dispositivos, cosa que se consigue con distintos tipos de mensajes propios del protocolo que abarca esta sección. Dichos mensajes proporcionan capacidades para registrar y/o invitar un usuario a una sesión, negociar los parámetros de una sesión, establecer una comunicación entre dos a más dispositivos y, por último, finalizar sesiones.

---

<sup>14</sup> IETF: Grupo de Trabajo de Ingeniería de Internet

## OPENCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multi núcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel

## SOCKETS

Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (*API, application programming interface*).

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como FTP<sup>15</sup>, Gopher<sup>16</sup>, o WWW).

Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

---

<sup>15</sup> FTP: 'Protocolo de Transferencia de Archivos'

<sup>16</sup> Gopher: es un servicio de Internet consistente en el acceso a la información a través de menús.

Para ello son necesarios los dos recursos que originan el concepto de socket:

- Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.
- Un par de números de puerto, que identifican a un programa dentro de cada computadora.

Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los programas que se denomina programa "cliente". El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa "servidor".

Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Las propiedades de un socket dependen de las características del protocolo en el que se implementan. El protocolo más utilizado es TCP (*Transmission Control Protocol*); una alternativa común a éste es UDP (*User Datagram<sup>17</sup> Protocol*).

Cuando se implementan con el protocolo TCP, los sockets tienen las siguientes propiedades:

- Son orientados a la conexión.
- Se garantiza la transmisión de todos los octetos sin errores ni omisiones.
- Se garantiza que todo octeto llegará a su destino en el mismo orden en que se ha transmitido.

Estas propiedades son muy importantes para garantizar la corrección de los programas que tratan la información.

El protocolo UDP es un protocolo no orientado a la conexión. Sólo se garantiza que si un mensaje llega, llegue bien. En ningún caso se garantiza que llegue o que lleguen todos los mensajes en el mismo orden que se mandaron. Esto lo hace adecuado para el envío de mensajes frecuentes pero no demasiado importantes, como por ejemplo, un *streaming* de audio.

---

<sup>17</sup> **Datagram:** es un paquete de datos que constituye el mínimo bloque de información en una red de conmutación

## Desarrollo

El proyecto se implemento de la siguiente manera:

### Instalación y configuración de Asterisk

Previamente para la instalación de Asterisk, necesitaremos instalar unas dependencias que serán necesarios para su instalación:

- aptitude install build-essential
- aptitude install linux-headers-`uname -r`

Ya teniendo las dependencias complementes instaladas, vamos a la instalación de Asterisk.

#### *Librerías Necesarias*

```
wget http://downloads.asterisk.org/pub/telephony/libpri/releases/libpri-1.4.12.tar.gz
```

```
wget http://downloads.asterisk.org/pub/telephony/libss7/releases/libss7-1.0.2.tar.gz
```

#### Asterisk

```
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-1.8.15.tar.gz
```

Una vez descargado todo, vamos a su desempaquetado:

```
tar -xzvf /usr/src/libpri-1.4.12.tar.gz
```

```
tar -xzvf /usr/src/libss7-1.0.2.tar.gz
```

```
tar -xzvf /usr/src/asterisk-1.8.15.0.tar.gz
```

**Pasamos a su instalación, es conveniente que sigan el mismo orden de instalación:**

Libpri

```
cd /usr/src/asterisk/libpri-1.4.12
```

```
make && make install
```

Libss7

```
cd /usr/src/asterisk/libss7-1.0.2
```

```
make && make install
```

Asterisk

```
cd /usr/src/asterisk/asterisk-1.8.15.0
```

```
./configure
```

```
make & make install
```

Escribimos el siguiente comando para cargar el demonio:

```
make config
```

Ejecutamos:

```
make samples
```

Para que nos cree los archivos de configuración básicos.

Ya instalado completamente Asterisk, vamos con la configuración de los clientes:

## Clientes y configuración

A la hora de añadir clientes o como se llaman en este caso "extensiones", debemos de tocar 2 archivos fundamentales que son "sip.conf" y "extensions.conf", vamos con el primero:

*Archivo sip.conf*

```
nano /etc/asterisk/sip.conf
```

Ya dentro del archivo, añadimos al final del mismo lo siguiente:

[200] ----> Extensión o número el cual usará dicho cliente para comunicarse con el resto.

type=friend

secret=password ----> Contraseña que usaremos para configurar el cliente..

qualify=yes -----> Yes o No, si queremos que el cliente esté disponible.

nat=no -----> Si el cliente estuviese detrás de una red distinta por nat, cambiaríamos a yes.

host=dynamic -----> Dynamic si dicha configuración puede ser usado por varios clientes.

canreinvite=no

mailbox=..... -----> Dirección para el contestador si deseamos que el cliente disponga de uno.

Como se puede ver en la figura 1, también he creado un segundo cliente con extensión 300 para la prueba.

```
[general]

context=default
bindport=5060
srlookup=yes
language=en
disallow=all
allow=alaw
allow=ulaw
allow=gsm

[200]

type=friend
host=dynamic
secret=200
context=user

[300]

type=friend
host=dynamic
secret=300
context=user
```

*Fig. 1 configuración del archivo sip.conf*

## Archivo extensions.conf

nano /etc/asterisk/extensions.conf

En este archivo (véase la figura 2), también nos vamos al final del archivo y añadimos lo siguiente:



```

manuel@MICHOMPU: /etc/asterisk
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.2.6 Fichero: extensions.conf Modificado

[general]

static=yes
writeprotect=no
autofallthrough=yes
priorityjumping=no
clearglobalvars=no

[user]
exten => 200,1,Dial(SIP/200,20,r)
exten => 200,n,Hangup()

[user]

exten => 300,1,Dial(SIP/300,20,r)
exten => 300,n,Hangup()

^G Ver ayuda ^O Guardar ^R Leer Fich ^Y Pág Ant ^K CortarTxt ^C Pos actual
^X Salir ^J Justificar ^W Buscar ^V Pág Sig ^U PegarTxt ^T Ortografía

```

Fig. 2 configuración del archivo extensions.conf

Ya solo nos quedaría que asterisk vuelva a leer de nuevo los ficheros, como se muestra en la figura 3.

/etc/init.d/asterisk reload

```

manuel@MICHOMPU: /etc/asterisk
Archivo Editar Ver Buscar Terminal Ayuda
sip.conf skinny.conf smdi.conf
manuel@MICHOMPU:/etc/asterisk$ nano sip.conf
manuel@MICHOMPU:/etc/asterisk$ su
Contraseña:
root@MICHOMPU:/etc/asterisk# nano sip.conf
root@MICHOMPU:/etc/asterisk# nano extensions.
extensions.ael extensions.conf extensions.lua
root@MICHOMPU:/etc/asterisk# nano extensions.conf
root@MICHOMPU:/etc/asterisk# asterisk -vvvr
Asterisk 1.8.13.1-dfsg1-3+deb7u3, Copyright (C) 1999 - 2012 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public License version 2 and other licenses; you are welcome to redistribute it under certain conditions. Type 'core show license' for details.
=====
== Parsing '/etc/asterisk/asterisk.conf': == Found
== Parsing '/etc/asterisk/extconfig.conf': == Found
Connected to Asterisk 1.8.13.1-dfsg1-3+deb7u3 currently running on MICHOMPU (pid = 2584)
Verbosity was 0 and is now 3
MICHOMPU*CLI>

```

Fig. 3 La imagen anterior nos muestra corriendo Asterisk.

## Configuración de los sofphone (Zoiper)

Esta es la pantalla principal de Zoiper (figura 4), para configurarlo y que pueda funcionar como extensión de asterisk se necesita hacer lo siguiente:



Fig. 4 Pantalla Inicial de Zoiper, en ella eliges el idioma, crear una nueva cuenta etc.

- creamos una nueva cuenta SIP



Fig. 5 En esta imagen se tiene que elegir el tipo de cuenta, en nuestro caso es SIP.

- llenamos los campos que nos piden (véase la figura 5), los cuales son nombre de la extensión, dominio, y la contraseña, que implementamos en la configuración de Asterisk.



Fig. 5 En la imagen anterior se muestra los campos que se tienen que llenar los cuales corresponden al numero de extensión, contraseña y dirección del servidor.

Listo ya tenemos configurada nuestras extensiones, ya están listas para usarse.

Ahora veremos funcionando completamente nuestras extensiones y Asterisk (véase la figura 6).

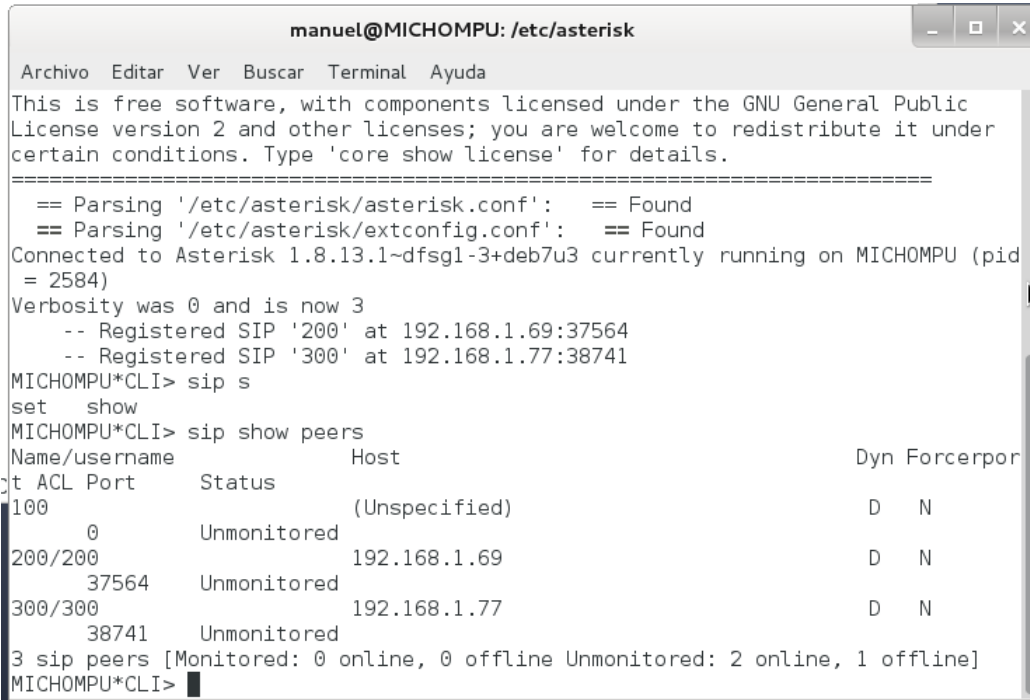


Fig. 6 Ejemplo de Asterisk con las extensiones 200 y 300 registradas con éxito.

```
manuel@MICHOMPU: /etc/asterisk
Archivo Editar Ver Buscar Terminal Ayuda
200/200      192.168.1.69      D N
  37564      Unmonitored
300/300      192.168.1.77      D N
  38741      Unmonitored
3 sip peers [Monitored: 0 online, 0 offline Unmonitored: 2 online, 1 offline]
== Using SIP RTP CoS mark 5
-- Executing [300@user:1] Dial("SIP/200-00000000", "SIP/300,20,r") in new st
ack
== Using SIP RTP CoS mark 5
-- Called SIP/300
-- SIP/300-00000001 is ringing
== Spawn extension (user, 300, 1) exited non-zero on 'SIP/200-00000000'
== Using SIP RTP CoS mark 5
-- Executing [200@user:1] Dial("SIP/300-00000002", "SIP/200,20,r") in new st
ack
== Using SIP RTP CoS mark 5
-- Called SIP/200
-- SIP/200-00000003 is ringing
-- Got SIP response 486 "Busy Here" back from 192.168.1.69:37564
-- SIP/200-00000003 is busy
== Everyone is busy/congested at this time (1:1/0/0)
-- Executing [200@user:2] Hangup("SIP/300-00000002", "") in new stack
== Spawn extension (user, 200, 2) exited non-zero on 'SIP/300-00000002'
MICHOMPU*CLI>
```

Fig. 7 prueba de una llamada entre los 2 sofphone.

Ya solo faltaba comprobar la conexión entre las dos extensiones, y en la pantalla de arriba nos lo describe, donde primero la extensión 200 hace una llamada a la extensión 300 y después viceversa la extensión 300 hace una llamada a la extensión 200.

El siguiente paso es capturar los paquetes sip de alguna llamada, y para esto usamos SIPDUMP, el cual se instala y se usa como se explica a continuación:

### Instalación de SIPDUMP

- Solo basta con hacer:

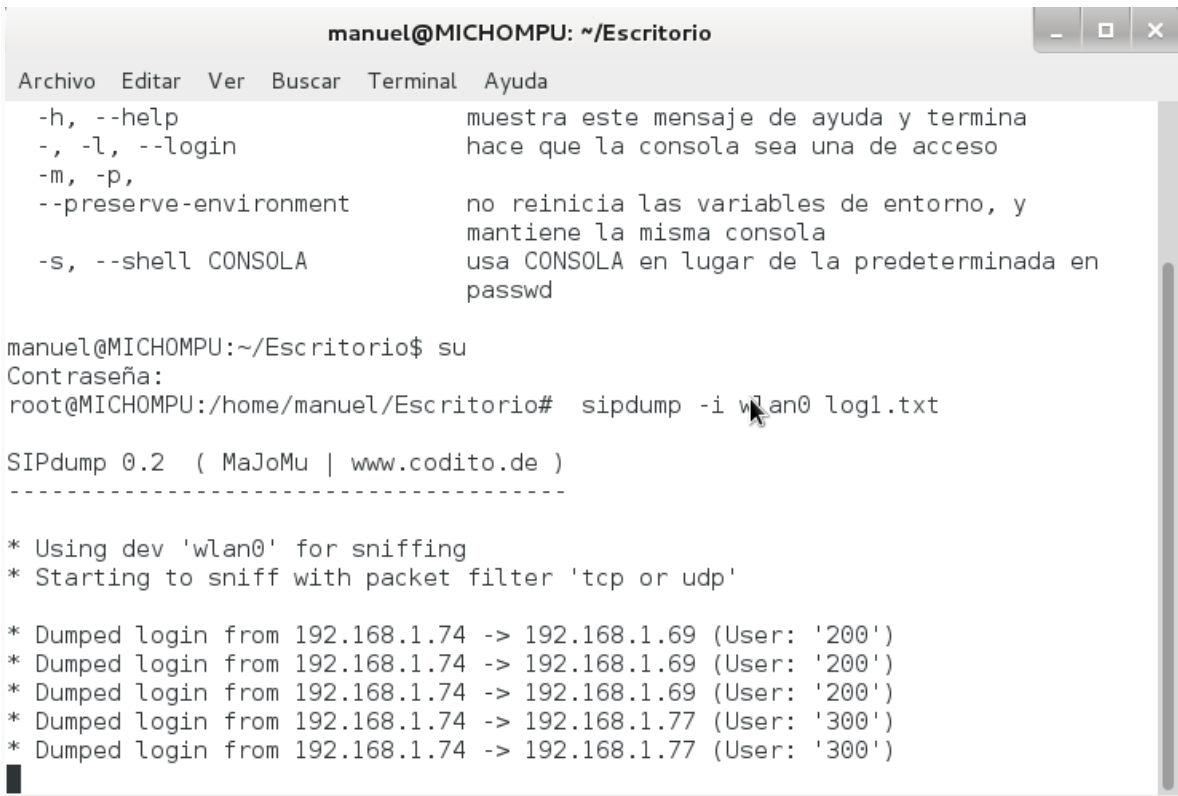
```
apt-get install sipcrack
```

- El siguiente es el hacer un filtro

```
Sipdump -i wlan0 log1.txt
```

Esto nos muestra lo siguiente:

Sipdump va mostrando cualquier cosa relacionada con paquetes SIP, como se muestra en la figura 8 y los almacena en un archivo de texto.



```
manuel@MICHOMPU: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
-h, --help                muestra este mensaje de ayuda y termina
-, -l, --login            hace que la consola sea una de acceso
-m, -p,
--preserve-environment    no reinicia las variables de entorno, y
                           mantiene la misma consola
-s, --shell CONSOLA       usa CONSOLA en lugar de la predeterminada en
                           passwd

manuel@MICHOMPU:~/Escritorio$ su
Contraseña:
root@MICHOMPU:/home/manuel/Escritorio# sipdump -i wlan0 log1.txt

SIPdump 0.2 ( MaJoMu | www.codito.de )
-----
* Using dev 'wlan0' for sniffing
* Starting to sniff with packet filter 'tcp or udp'

* Dumped login from 192.168.1.74 -> 192.168.1.69 (User: '200')
* Dumped login from 192.168.1.74 -> 192.168.1.69 (User: '200')
* Dumped login from 192.168.1.74 -> 192.168.1.69 (User: '200')
* Dumped login from 192.168.1.74 -> 192.168.1.77 (User: '300')
* Dumped login from 192.168.1.74 -> 192.168.1.77 (User: '300')
```

Fig. 8 En la imagen anterior se puede observar la ejecución de SIPdump.

Sipdump va almacenando los paquetes que existen entre una llamada que usa dicho protocolo, como se muestra en la figura 9 existe un paquete INVITE este paquete es el que nos interesa capturar para el siguiente paso de este proyecto, el SIPDUMP lo utilizamos con ese propósito, de capturar el paquete INVITE que es el que contiene la información que nos interesa para cumplir con el objetivo de nuestro proyecto.



```
log1.txt x
El archivo /home/manuel/Escritorio/log1.txt ha cambiado en el disco.
¿Quiere volver a cargar el archivo?

192.168.127.13"192.168.127.20"200"asterisk"REGISTER"sip:192.168.127.20;transport=UDP"344c8
192.168.127.13"192.168.127.20"200"asterisk"REGISTER"sip:192.168.127.20;transport=UDP"2b6b9
192.168.1.69"192.168.1.74"200"asterisk"INVITE"sip:300@192.168.1.74;transport=UDP"428da125"
192.168.1.69"192.168.1.74"200"asterisk"REGISTER"sip:192.168.1.74;transport=UDP"30f57573""
192.168.1.69"192.168.1.74"200"asterisk"REGISTER"sip:192.168.1.74;transport=UDP"0b0eea2a""
192.168.1.77"192.168.1.74"300"asterisk"REGISTER"sip:192.168.1.74;transport=UDP"5a770d48""
192.168.1.77"192.168.1.74"300"asterisk"REGISTER"sip:192.168.1.74;transport=UDP"73204131""
```

Fig. 9 Sipdump va almacenando los paquetes sip en un archivo txt.

El siguiente paso es explicar paso a paso el código de nuestros programas:

## Código Script bash

```
#!/bin/bash
```

```
cat /dev/null > /home/manuel/Escritorio/log1.txt
```

En primera instancia explicaremos nuestro script, donde la primera línea así no lo indica, que se trata de un bash, el cual va a llevar el control de archivos auxiliares a nuestro programa.

Con la segunda línea borramos el contenido del archivo log1.txt, es el que creamos con el sipdump, se vacía el contenido del archivo para recibir nuevos datos, en el caso que haya una nueva llamada.

```
while true;
do
while read line
do
echo -e "$line\n" | grep -o '".*"' >> creacion.txt
done < /home/manuel/Escritorio/log1.txt
```

El siguiente código es uno de lo mas importantes, ya que recorre línea por línea el archivo que creamos con sipdump, el log1.txt y lo empieza a limpiar, eso lo realizamos con grep para que nada más nos muestra la línea con la información que trae el paquete INVITE, la información filtrada la guardamos en un archivo llamado creación.txt, este archivo es el que vamos a utilizar, para desplegar su contenido en el streaming de video.

```
f1=creacion.txt;
if [ -s $f1 ]
then
sleep 15
```

```
    echo "borrado $f1"

cat /dev/null > creacion.txt
cat /dev/null > /home/manuel/Escritorio/log1.txt

fi

done
```

El ultimo pedazo de código es una condición en el cual revisa si el archivo esta vacio o no, si el archivo creación.txt tiene contenido después de cierto tiempo lo borra para almacenar la información de la siguiente llamada.

Por último explicaremos a detalle el código de nuestro proyecto:

### Explicación del código

Antes de empezar a explicar el código cabe mencionar que en parte del código utilizamos la biblioteca Opencv, entonces primero instalaremos esta biblioteca para que la podamos usar:

- Primero nos aseguramos de que todo en el sistema esta actualizado

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Ahora necesitaremos instalar todas las dependencias, como el soporte para leer y escribir imágenes, dibujar en la pantalla etc.

Este paso es muy sencillo solo hay que ejecutar lo siguiente en la terminal:

```
Sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev
```

Ahora vamos a obtener el codigo fuente de Opencv 2.4.2:

```
Wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix
```

```
Tar -xvf Opencv-2.4.2.tar.bz2
```

Cd Opencv-2.4.2

Vamos a generar el Makefile usando cmake. Aquí podemos definir que partes de Opencv queremos compilar. Como vamos a usar Python, TBB, OpenGL, Qt, trabajar con videos, etc. debemos establecer todos esto aquí. Ejecuta la siguiente línea en la terminal para crear el Makefile apropiado.

Mkdir build

Cd build

```
Cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON
```

Llegados a este punto, estamos listos para compilar e instalar Opencv 2.4.2

Make

Sudo make install

Ahora tenemos que configurar Opencv. Abre el fichero opencv.conf con algún editor por ejemplo:

```
Sudo nano /etc/ld.so.conf.d/opencv.conf
```

- Añade la siguiente línea al final del fichero y guardalo

```
/usr/local/lib
```

- Ejecuta la siguiente línea para para configurar la librería

```
Sudo ldconfig
```

- Abre /etc/bash.bashrc y añade las siguientes líneas al final del archivo y guardalo

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
```

```
Export PKG_CONFIG_PATH
```

- Cierra la terminal y reinicia.

Listo ya tenemos lista la librería que vamos a utilizar, ahora a explicar el código a detalle.



```
#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <cv.h>

#include <highgui.h>
```

Empezamos con las librerías, algunas de ellas ya se nos hacen conocidas, pero hay otras que pertenecen a opencv, las cuales nos permitirá manejar el video.

```
int main(int argc, char *argv[])
{
    CvCapture *dispositivo = NULL;

    IplImage *fotograma = NULL;

    int tecla = 0, ndispositivo = CV_CAP_ANY;

    int bandera=1;

    char caracter[50];

    int contador=0;

    int contador2=0;

    int c;

    int lb;

    int la;

    CvFont font;

    CvPoint pt1;

    cvInitFont( &font, CV_FONT_VECTOR0, 0.5, 0.5, 0, 2.0, CV_AA); //Inicializamos el código fuente
```

Comenzamos con nuestro main, declarando variables que se utilizaran mas adelante, unas de las variables más extrañas son: CvCapture \*dispositivo, IplImage \*fotograma , ndispositivo = CV\_CAP\_ANY; estas variables se usan para guardar cada fotograma que capturaremos con nuestra webcam, al final del código definimos los parámetros para el texto que aparecerá el en el flujo de video.

```

/* Inicializamos el dispositivo de captura y comprobamos el estado */
    printf("Inicializando dispositivo de captura #%%d\\n", ndispositivo);

dispositivo = cvCaptureFromCAM(ndispositivo);

    if (dispositivo == NULL) {
fprintf(stderr, "Error inicializando dispositivo de captura #%%d\\n", ndispositivo);
        return 1;
    }

/* Creamos una ventana para mostrar el contenido de la webcam */
    printf("Creando ventana\\n");

    cvNamedWindow("Fotograma", CV_WINDOW_AUTOSIZE);

```

El código anterior empieza a capturar frames de la webcam y las guarda en la variable dispositivo, en caso de que la variable dispositivo es igual a null quiere decir que no hay webcam conectada y te arroja un mensaje de error. En la última parte del código, la función cvNamedWindow() crea una ventana para mostrar lo capturado con la webcam.

```
pt1.x = 220;
```

```
pt1.y = 30;
```

```
FILE *archivo;
```

```
FILE *archivo2;
```

```
FILE *archivo3;
```

```
FILE *archivo4;
```

```
printf("Entrando en el bucle principal (ESCh para salir)\\n");
```

```
while(1) {  
  
contador=contador+1;  
  
fotograma = cvQueryFrame(dispositivo);
```

Bueno empezamos con el código interesante del proyecto, lo podemos ver claramente que no se trata de una cosa muy complicada, en la parte de arriba declaramos, las coordenadas donde aparecerá el texto, seguido declaramos punteros a ficheros que utilizaremos para manipular las banderas, las cuales nos ayudaran a quitar y poner el texto, después comienza nuestro ciclo infinito, es infinito ya que queremos que el flujo de video sea indefinido, lo que sigue es un contador, este nos servirá para llevar un tipo de timer, que nos auxiliara en poner el texto cuando sea necesario, por ultimo empezamos a capturar con la webcam y todo lo capturado se va almacenando en la variable fotograma.

```
if(fotograma != NULL) {  
  
archivo2 = fopen("lec.txt","r");  
  
c=fgetc(archivo2);  
  
if(c=='b')  
{  
  
bandera=0;  
  
}  
  
else{  
  
bandera=1;  
  
}  
  
if(bandera==0){  
  
archivo = fopen("creacion.txt","r");  
  
fflush(archivo);  
  
fgets(caracter,50,archivo); //obtenemos caracteres del archivo
```

```
cvPutText(fotograma,caracter, pt1, &font,CV_RGB(255,190,44) ); //ponemos el texto en
el video
```

```
fclose(archivo);
```

```
}
```

```
fclose(archivo2);
```

```
}
```

En esta parte del código entramos en la parte lógica del programa, en primera instancia comienza con una condición que nos dice que si tenemos video que mostrar has lo siguiente: abrimos el archivo llamado lec.txt, obtenemos el carácter que se encuentra en ese archivo, si el carácter es diferente de “b” pone a la variable bandera en 1 y no realiza nada sigue grabando, pero no realiza ninguna acción, sin embargo, si el carácter es una “b”,pone a la variable bandera en 0 y entra en una segunda condición, la cual nos indica lo siguiente:

Si la variable bandera es igual a 0, abrimos el archivo llamado creación.txt, obtenemos los caracteres que contiene el archivo y los ponemos en flujo de video con la función cvPutText(), finalmente cerramos los archivos abiertos.

```
cvShowImage("Fotograma", fotograma);
```

```
tecla = cvWaitKey(100);
```

```
if(tecla == 27)
```

```
break;
```

El código anterior nos muestra en pantalla el flujo de video, y en la parte de abajo nos indica que si la tecla que presionamos es “esc” se interrumpe el flujo y se termina el programa.

```

archivo2 = fopen("lec.txt","r+");
archivo3 = fopen("poner.txt","r");
lb=fgetc(archivo3);
if(contador==20){
fputc(lb,archivo2);
fclose(archivo2);
fflush(archivo2);
}
fclose(archivo3);
fflush(archivo3);
archivo4= fopen("obtenera.txt","r");
la=fgetc(archivo4);
if(contador==100){
archivo2 = fopen("lec.txt","w");
fputc(la,archivo2);
fclose(archivo2);
}
fclose(archivo4);
fflush(archivo4);

```

En el penúltimo fragmento de código jugamos un poco con los archivos auxiliares que declaramos en un principio, y el código se desglosa de la siguiente manera:

Empezamos abriendo los archivos lec.txt y poner.txt, obtenemos el carácter que se encuentra en el archivo poner.txt y después de que nuestro timer llegue a 20 el carácter obtenido lo ponemos en el archivo lec.txt, cerramos nuestro archivo y enseguida abrimos otro archivo llamado obtenera.txt obtenemos el carácter que almacena dicho archivo y cuando nuestro timer llegue a 100, nuevamente abrimos el archivo lec.txt y le

sobrescribimos el carácter que obtuvimos, esto se realiza con la finalidad de que el texto que se pone en el flujo de video se quite transcurrido cierto tiempo, por ultimo cerramos los archivos correspondientes.

```
archivo = fopen("creacion.txt","r");  
fseek( archivo, 0, SEEK_END );  
if (ftell(archivo) == 0 ){  
contador=0;  
printf("Esta vacio\n");  
}  
else  
printf("tiene texto\n");  
fclose(archivo)  
} //ciERRA WHILE
```

Por último modificamos el archivo creación.txt y lo dejamos listo para que almacene los datos de una nueva llamada, primero lo abrimos, usamos la función fseek() para recorrer el cursor al principio del archivo, borramos su contenido e inicializamos el timer a 0, listo

```
printf("\nLiberando recursos\n");  
cvDestroyWindow("Fotograma");  
cvReleaseCapture(&dispositivo);  
return 0;  
}
```

Finalmente liberamos los recursos usados y cerramos nuestro main.

## Ejemplo del proyecto

Lo primero es correr Asterisk y Sipdump, como se muestra en la figura 10, podemos ver en la pantalla de la izquierda nuestras dos extensiones registradas con éxito, al igual en la pantalla de la derecha Sipdump empieza a registrar los movimientos de nuestras extensiones.

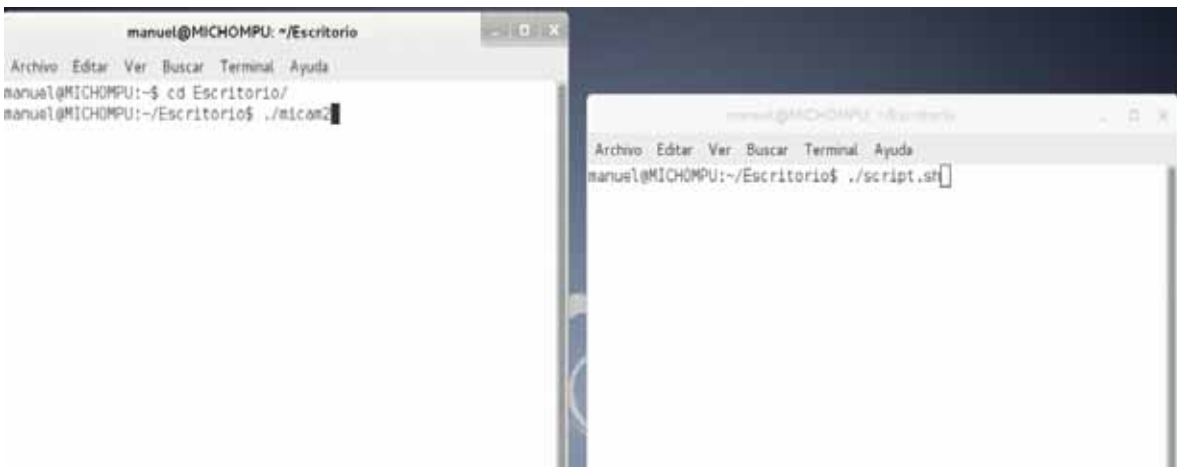


The image shows two terminal windows side-by-side. The left window is the Asterisk CLI, and the right window is a terminal running Sipdump. The Asterisk CLI shows the registration of two extensions, 300 and 200, with their respective ports and IP addresses. The Sipdump terminal shows the command being executed and the resulting log entries for the two extensions.

```
manuel@MICHOMPU:~/Escritorio
manuel@MICHOMPU:~$ cd Escritorio/
manuel@MICHOMPU:~/Escritorio$ su
root@MICHOMPU:/home/manuel/Escritorio# sipdump -i wlan0 log1.txt
SIPdump 0.2 ( MaJoMu | www.codito.de )
-----
* Using dev 'wlan0' for sniffing
* Starting to sniff with packet filter 'tcp or udp'
* Dumped login from 192.168.1.74 -> 192.168.1.67 (User: '300')
* Dumped login from 192.168.1.74 -> 192.168.1.67 (User: '300')
* Dumped login from 192.168.1.74 -> 192.168.1.69 (User: '200')
* Dumped login from 192.168.1.74 -> 192.168.1.69 (User: '200')
```

Fig. 10 La imagen anterior muestra la ejecución de Asterisk y SIPdump.

Corriendo perfectamente Asterisk y Sipdump, ahora tenemos que correr nuestros programas para que empiecen a funcionar nuestro programa, se deben correr de manera simultánea.



The image shows two terminal windows side-by-side. The left window is the Asterisk CLI, and the right window is a terminal running a script. The Asterisk CLI shows the command being executed, and the right window shows the script being executed.

```
manuel@MICHOMPU:~/Escritorio
manuel@MICHOMPU:~$ cd Escritorio/
manuel@MICHOMPU:~/Escritorio$ ./micar2

manuel@MICHOMPU:~/Escritorio
manuel@MICHOMPU:~/Escritorio$ ./script.sf
```

Fig. 11 se ejecutan los programas de manera simultánea.

Inicia el flujo de video.



*Fig. 12 Imagen que muestra al correr los programas.*

Cuando la extensión 200 hace una llamada aparece, se captura el paquete INVITE, el cual contiene la información que se despliega en pantalla (véase la figura 13).



*Fig. 13 En la imagen anterior se muestra el texto de extensión que realizo la llamada en este caso fue la extensión 200.*



De manera simultánea cuando la extensión 300 realiza una llamada.

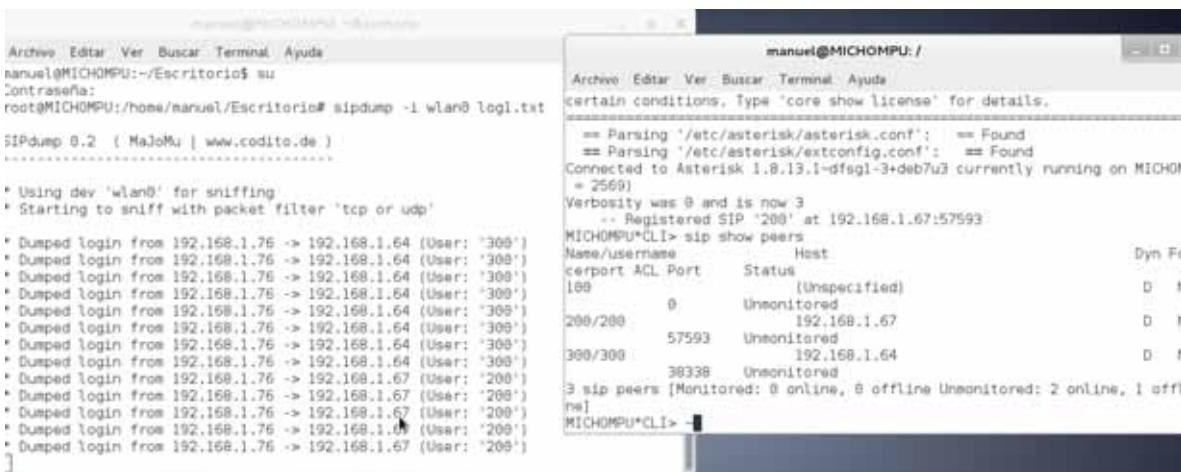


*Fig. 14 En la imagen anterior se muestra el texto de extensión que realizo la llamada en este caso fue la extensión 300.*

## Resultados

Finalmente llegamos a los resultados hasta este punto nuestro proyecto funciona de forma correcta como se puede ver en el ejemplo anterior, sin embargo va mas allá, ahora queremos que el video se pueda ver desde otra PC. Se hicieron pruebas con tecnologías de terceros, como lo es ffmpeg, tratar de verlo desde un navegador utilizando Html5 y Java Script, pero no se obtuvo ningún éxito, se llego a la conclusión que la forma más segura es utilizando sockets, solo que un pequeño detalle por cuestiones de tiempo ya no fue posible hacer la prueba corriendo un cliente desde otra PC pero si se hizo desde la misma PC, como a continuación se muestra y se explica a detalle.

En la figura 15, se muestra corriendo lo que sería la base de nuestro proyecto, Asterisk funcionando, se puede observar que ya tiene registradas las extensiones de manera exitosa, en la pantalla de nuestra derecha se muestra nuestro sipdump, el cual nos va ayudar a capturar todo el trafico SIP.



The image shows two terminal windows. The left window is running 'sipdump -i wlan0 log1.txt' and shows a list of captured SIP login events from 192.168.1.76 to 192.168.1.64 and 192.168.1.67. The right window is running 'sip show peers' and displays the following table:

Name/username	Host	Dyn	For
100	(Unspecified)	D	N
200/200	192.168.1.67	D	N
300/300	192.168.1.64	D	N

Below the table, it shows '3 sip peers [Monitored: 0 online, 0 offline Unmonitored: 2 online, 1 offline]'. The terminal prompt is 'MICHOMPU\*CLI>'.

Fig. 15 Ejecución de Asterisk y SIPdump.

La imagen anterior nos muestra la preparación de nuestros programas para arrancarlos, estos programas son el script, que ya habíamos descrito y nuestro socket\_server (véase el apéndice A) y el socket\_cliente (véase el apéndice A). Teniendo ya listo todo solo basta con correrlos, aquí cabe menciononar una que el orden en que los corre es importante se sugiere correr primero el script, segundo el socket\_server y por último el socket\_cliente, para obtener el siguiente resultado:

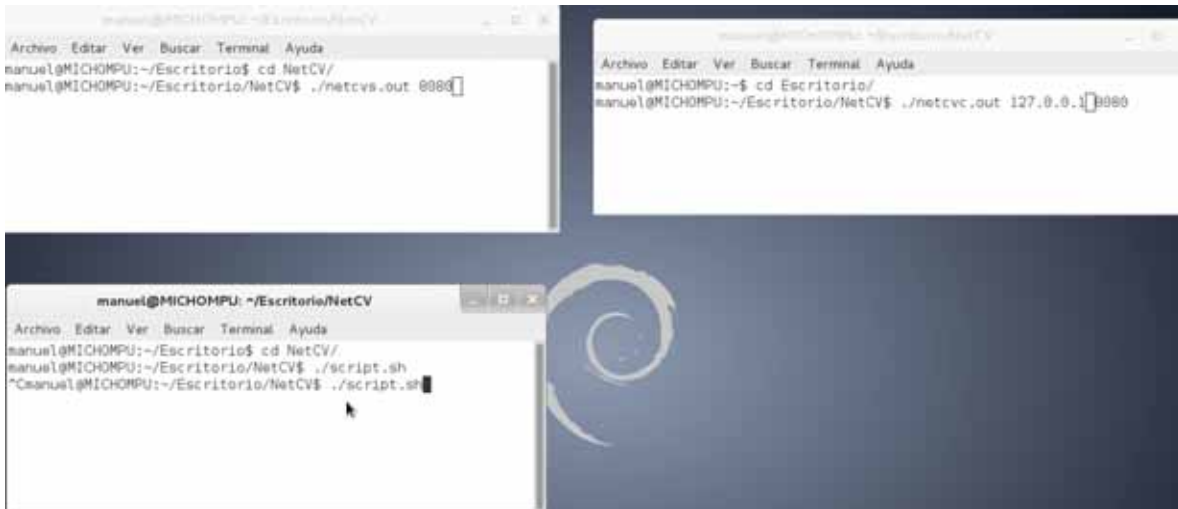


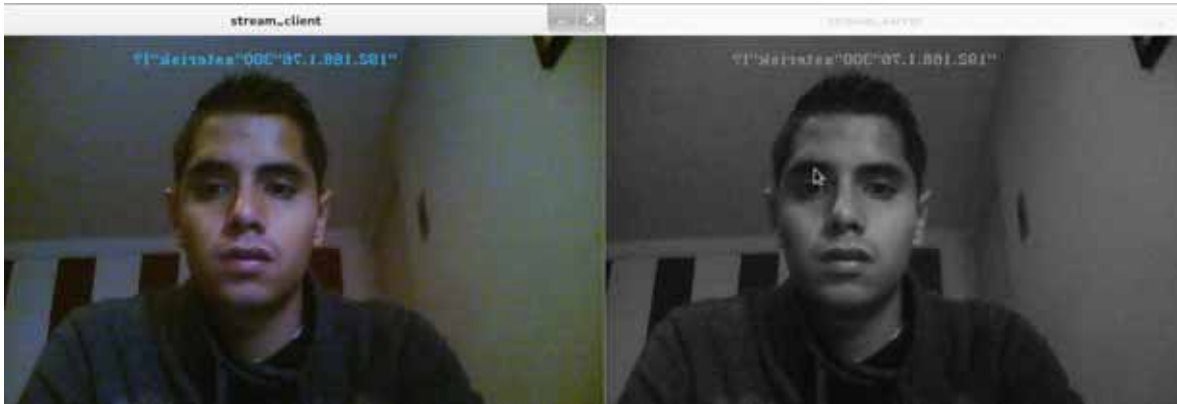
Fig. 16 Preparación para ejecutar el script, el socket\_cliente y socket\_server.

Hasta aquí todo marcha a la perfección ya tenemos nuestro flujo de video (véase la figura 17), solo queda desplazar el texto en tiempo real sobre el video, al realizar una llamada de una extensión a otra obtuvimos el siguiente resultado:



Fig. 17 La imagen anterior nos muestra el resultado de la ejecución de nuestros programas.

En este punto de nuestro proyecto nos llevamos una sorpresa, el texto obtenido se desplaza al revés, se sospecha que este resultado se obtuvo por un mal parámetro de la función Puttext(), por cuestiones de tiempo ya no se pudo arreglar pero el texto es correcto el único detalle es que lo despliega al revés como lo podemos observar en la figura 18.



*Fig. 18 Al hacer una llamada la extensión que realiza la llamada se despliega sobre el flujo de video que está en curso.*

## Conclusión

Con todo lo explicado y desarrollado anteriormente podemos concluir que es posible añadir texto en tiempo real a un streaming de video sobre una arquitectura IMS, y con ello se puede resolver la problemática que se tenía, que pasa cuando estás viendo un video, de repente suena tu teléfono, pero no lo tienes cerca de ti, este proyecto resolvió dicho problema ya que te despliega en pantalla y en tiempo real, la persona que te está llamando, se realizaron pruebas con proyectos ya hechos como ffmpeg, en donde no tuvimos éxito se pudo visualizar el flujo de video pero no se pudo añadir el texto en tiempo real, también se trato de utilizar Html5 en compañía de Java Script sin ningún éxito, con esto pudimos comprobar que este programa no agrega texto en tiempo real, por otro lado se aprendió la configuración básica de Asterisk, así como Sipdump, herramienta la cual desconocía por completo, al saber de ella, investigamos sobre el funcionamiento de esta herramienta, la pusimos en práctica para capturar el tráfico de llamadas SIP, cabe mencionar que las herramientas utilizadas fue software totalmente nuevo para mí, se utilizo la librería Opencv, para las cuestiones del manejo del flujo de video.

Se obtuvo nuevos conocimientos de programación al utilizar la librería Opencv, así como de la telefonía VoIP, con la investigación de documentación de Asterisk, gran problemática del proyecto se resolvió utilizando una lógica muy sencilla, les comento esto ya que en ocasiones tratamos de buscar una solución verdaderamente compleja, siendo que el problema se puede resolver con algo mucho más sencillo, así que a estas alturas les puedo aconsejar que en ocasiones no es necesario quemarse de mas la cabeza, empieza buscando una solución sencilla y no al revés.

Finalmente para la realización de este proyecto, fue utilizar sockets, los cuales nos dieron buenos resultados, nos permitieron cumplir con los objetivos planteados, solo tuvimos un problema el texto se despliega al revés, como lo pudimos observar más arriba en el documento, pero el problema radica en un valor de un parámetro de la función PutText(), ya que esta fue modificada a la hora de utilizar sockets

## Bibliografía

[1] **Learning OpenCV: Computer Vision with the OpenCV Library**, Gary Bradski (Autor), Adrian Kaehler (Autor).

[2] **Rec UIT-T Y.2021, “Subsistema multimedia IP para las redes de la próxima generación”**. Septiembre 2006.

[3] **Simón Znaty, Jean-Louis Dauphin, Roland Geldwerth, “IP Multimedia Subsystem: Principios y Arquitectura”**. <http://www.efort.com>

[4] **Alberto Hernández, Manuel Álvarez-Campana, Enrique Vázquez and Vicente Olmedo, “The IP Multimedia Subsystem (IMS). Quality of service and performance simulation”** Universidad Politécnica de Madrid. Julio 2007

[5] **Gonzalo Camarillo and Miguel A. García-Martín, “The 3G IP Multimedia Subsystem”**, Wiley 2004.

[6] **VoIP [en línea]: Recursos VoIP- Voz sobre IP: Telefonía IP**.  
<<http://www.recursosvoip.com/>>

[7] **Programación De Socket Linux**, Sean Walton(PRENTICE-HALL)

[8] **VoIP y Asterisk redescubriendo la telefonía**

[9] **SIP: Understanding the Session Initiation Protocol**, Alan B. Johnston, Artech House, Enero 2001, 228 páginas

## Apéndice A

**Para compilar los socket cliente y servidor se tienen que realizar los siguientes pasos**

**El programa netcvs.c corresponde al socket del servidor su compilación se realiza con la siguiente línea**

```
manuel@MICHOMPU:~/Escritorio/NetCV$ g++ netcvs.cpp -o netcvs.out -I/opt/local/include -L/opt/local/lib -lopencv_core -lopencv_highgui -lopencv_imgproc
```

**Después para su ejecución se realiza con la siguiente línea**

```
manuel@MICHOMPU:~/Escritorio$ cd NetCV/  
manuel@MICHOMPU:~/Escritorio/NetCV$ ./netcvs.out 8080
```

**El programa netcvc.c corresponde al socket cliente su compilación se realiza con la siguiente línea**

```
manuel@MICHOMPU:~/Escritorio/NetCV$ g++ netcvc.cpp -o netcvc.out -I/opt/local/include -L/opt/local/lib -lopencv_core -lopencv_highgui -lopencv_imgproc
```

**Para su ejecución se realiza con la siguiente línea**

```
manuel@MICHOMPU:~$ cd Escritorio/  
manuel@MICHOMPU:~/Escritorio/NetCV$ ./netcvc.out 127.0.0.1 8080
```