Laboratorio 2

Thenison Encina *2803016-9*
Daniel Romero *2873039-k*

Valparaíso, 2 de mayo de 2016

1. Introducción

Como se vio en el ramo un socket es una puerta entre el proceso aplicación y el protocolo de transporte de extremo a extremo, son locales al host y son creados por cada aplicación, con la función de ser una interfaz a través de la cual el proceso aplicación puede recibir y enviar mensajes desde/hacia otro proceso aplicación.

La interfaz de sockets facilita la comunicación entre procesos, programando dicha comunicación de manera similar a como se maneja cualquier dispositivo de entrada salida, ya que independiza la localización de los procesos.

Un Socket permite implementar una arquitectura cliente/servidor para la que principalmente podemos mencionar las necesidades básicas para su implementación en TCP/IP:

1. Un par de Direcciones IP de los host.
2. Un par de número de puerto que representa al proceso corriendo en cada host. endenumerate

   Para el proceso de programación de los sockets existen dos tipos de aplicaciones cliente-servidor que se pueden generar:

   - Estándar RFC: Cuando se desea crear una aplicación que funcione con algún estándar definido por RFC.
   - Aplicaciones propietarias: están tiene la característica de que no se rigen por algún RFC existente y se debe tener cuidado en no utilizar puertos definidos en RFC.

   Además para la programación de los socket se debe elegir si usar el protocolo TCP u UDP.

2. Desarrollo

2.1. Programación con sockets

Para realizar nuestra tarea se utilizo el lenguaje Python ya que proporciona un marco para programar servidores de red que simplifica gran parte de la programación del socket y sus implementaciones propiamente tal y nos permite concentrarnos en la lógica del programa.

El marco (framework) a utilizar lleva por nombre socket el cual es una biblioteca estándar que define diferentes clases de conecciones que se pueden utilizar en este marco, las cuales son: TCPServer, UDPServer, UnixStreamServer y UnixDatagramServer, estas dos últimas deben ser utilizadas en plataformas Unix.

Las subrutinas de sockets toman como parámetros el tipo de socket y su protocolo. Las aplicaciones deben especificar estos para indicar el estilo de comunicación deseada.

Para crear código del cliente se requieren los siguientes pasos:

a) Importamos librerias

```
import socket
import time
import sys
import os
```

b) Se inicia mediante la creación de un socket TCP / IP y su conección.

```
def connection():
#Crea la concexion con el servidor
server = "10.10.14.3"
port = 9999

skt = socket.socket()
try:
        print "CONECTANDO..."
        skt.connect((server, port))
        print "CONEXION ESTABLECIDA"
        return skt
except Exception, e:
        print "ERROR DE CONEXION"
```

c) También de desconección

```
def disconnect(skt):
# Cierra la conexion del socket
try:
        print "DESCONECTANDO..."
        exit = sendMessage(skt, "EXIT")
        skt.close()
        print "CONEXION CERRADA"
except Exception, e:
        print "ERROR DE DESCONEXION"
```

d) Luego se debe instanciar la conección.

```
def initConnection(skt):
# Inicia la conexion con el servidor
try:
        print "INICIANDO..."
        #response = sendMessage(skt, "22")
        print "CONEXION INICIADA"
        #return response
except Exception, e:
        print "ERROR AL INICIAR"
```

e) Finalmente se instancia una de las clases para mensajes, pasando como parámetro el mensaje.

```
def sendMessage(skt, request):
#Envia un request
try:
      skt.send(request)
      response = skt.recv(BUFFER_SIZE)
      return response
except Exception, e:
      print "NO SE PUDO ENVIAR EL MENSAJE"
```

f) El programa Main consiste en:

```
# ===
# MAIN PROGRAM
# ===

skt = connection()
response = initConnection(skt)
response = sendMessage(skt, "22")
rp = skt.recv(BUFFER_SIZE)
print "respuesta: ", rp

response = sendMessage(skt, "Thenison Encina, Daniel Romero")
rp = skt.recv(BUFFER_SIZE)
print "respuesta: ", rp
```

Como se apreciara más adelante, los mensajes enviado son :

```
response = sendMessage(skt, "22")
response = sendMessage(skt, "Thenison Encina, Daniel Romero")
```

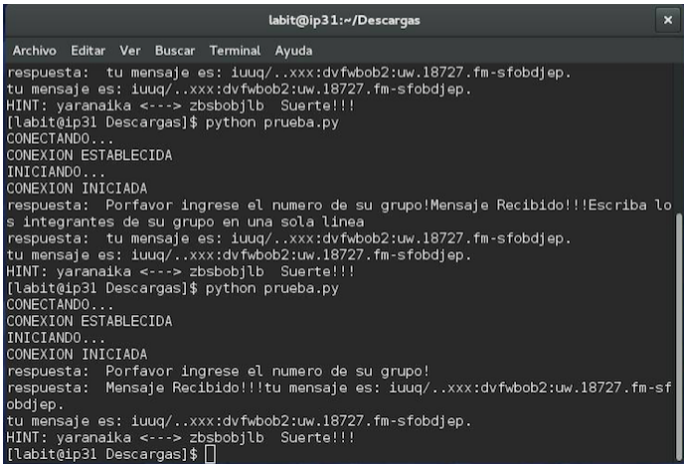
Ya que son parte de la interacción solicitada en el Lab.

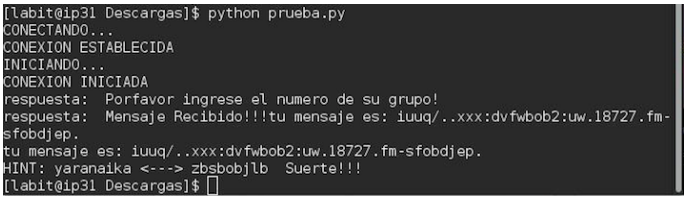
2.2. Resultados

Una vez establecida la conexión. El programa seguira el siguiente orden.

a) Mensaje de Bienvenida
b) Mensaje de número de grupo
c) Mensaje de Integrantes del Grupo
d) Mensaje Encriptado
e) Mensaje Encriptado
f) Pista

El resultado de esto se puede apreciar en la siguiente imagen:





a) Conexion iniciada
b) Porfavor ingrese el numero de su grupo!
c) Escriba los integrantes de su grupo en una sola linea
d) tu mensaje es: iuuq/..xxx:dvfwbob2:uw.18727.fm-sfobdjep.
e) tu mensaje es: iuuq/..xxx:dvfwbob2:uw.18727.fm-sfobdjep.
f) HINT: yaranaika $<--->$ zbsbobjlb Suerte!!

Del hint podemos obtener que para desencriptar el debemos remplazar el las letras por la letra siguiente en el abecedario.

Ej:
B$->$A

C− >B
D− >C

Hint:
Y− >Z
A− >B
R− >S

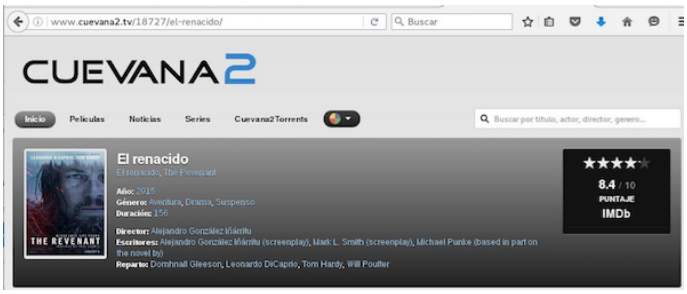
pero en verdad la pista nos dice que reemplacemos por la anterior:

Z− >Y
B− >A
S− >R

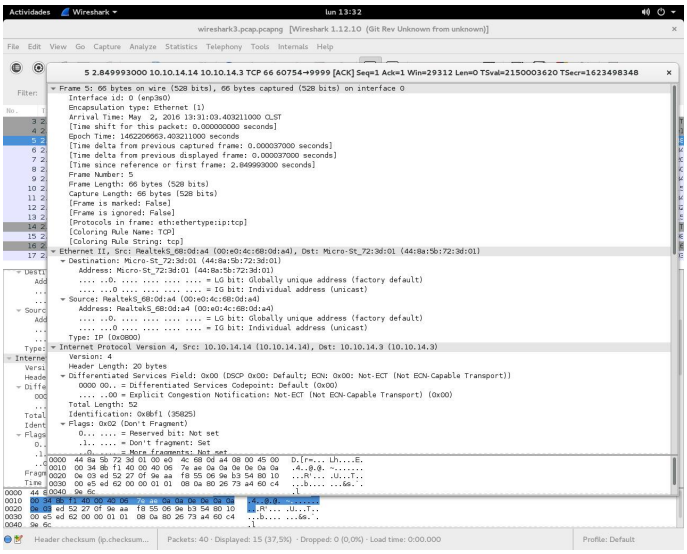
Al desencriptar el mensaje obtenemos :

http://www.cuevana2.tv/18727/el-renacido/

que es



Finalmente, se puede comprobar con WIRESHARK que al enviar mensajes al servidor, estos se comunican.



Filtrando un poco más los paquetes capturados :