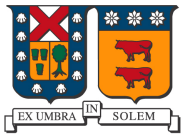


UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Redes de computadores
Arquitectura Cliente - Servidor

Oscar Lizama 201103043-5

Geordy Kindley 201151032-1

Javier Ignacio Jeria Morales 201373038-8

Profesor: Agustín Gonzales
Fecha: Vierness, 1 de Julio del 2016

1 Resumen

Trata sobre arquitectura de modelo Cliente - Servidor y su utilidad en la internet. El tema se aborda desde tres aristas. En primer lugar se explica la teoría asociada que hay detrás de este modelo. En segundo lugar trata acerca de como implementar el modelo utilizando como lenguaje Python y en último lugar, pero no menos importante, analizar los resultados obtenidos.

2 Introducción

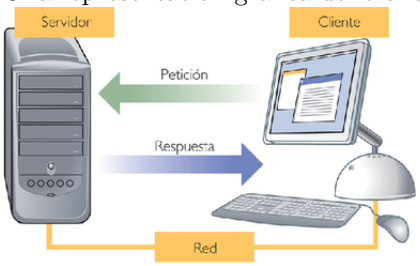
Dentro del esfuerzo humano a traves de la tecnología que día a día perfecciona, esta el persistente intento de máximizar una comunicación. Es aquí, donde consideramos como una de las ramas por excelencia preocupadas del asunto, **Redes de computadores**.
En particular, este informe trata sobre **arquitectura Cliente - Servidor**, el cual es un modelo de una aplicación distribuida en el cual se basa en dos actores: Uno con rol de proveedor de recursos y otro con rol consultor sobre los recursos. Esto será visto en detalle teorícamente y se hará un ejemplo practico de como se implementa utilizando como lenguaje Python v2.7 utilizando la libreria socket. A continuación se le invita a continuar la lectura del informe.

3 Teoría sobre arquitectura Cliente- Servidor

Como se anticipo anteriormente, dos actores son los fundamentales. El **Cliente** y el **Servidor**.

- **Cliente**: Programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo.
- **Servidor**: Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP y junto con otros programas de aplicación. El servidor comienza su ejecución antes de comenzar la interacción con el cliente.

Una representación grafica de lo anterior señalado podría ser lo siguiente:



Existen otros modelos de los cuales se puede comparar, como por ejemplo `peer to peer`. Sin embargo, la gran mayoría de las aplicaciones que usa en la cotidianeidad implementan este modelo cliente-servidor. Las ventajas de este modelo respecto de otras posibles arquitecturas de red serían:

- Centralizado
- Es de Fácil mantención
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.

Por otro lado, la mayor desventaja es que a mayor número de clientes, más problemas para el servidor, debido a la congestión en el trafico que se generá.

A continuación se presentará de modo breve un modo de implementar de modo simple un modelo cliente-servidor.

4 En la practica

Para lograr implementar una arquiectura Cliente - Servidor se deben crear las aplicaciones por separado.Esto permite que ambos no necesariamente sean programados en el mismo lenguaje, sin embargo, aquí se presentará la explicación utilizando Python v2.7 para realizar ambas aplicaciones, pues este es el lenguaje que se enseña en el curso que es requisito del ramo presente.
A continuación las lineas principales del código del Servidor. Tener presente que no es el código completo, pues la idea principal es abstraer y con ello simplificar la compresión del mismo.
Creando un socket TCP/IP:

```
import socket
import sys

# Creando el socket TCP/IP
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Para asociar un socket a una dirección de servidor se utiliza el método bind(). En este caso, la dirección es un ip (dirección actual) y el número de puerto es 9999.

```
# Enlace de socket y puerto
server_address = ('192.168.1.100', 9999)
print >>sys.stderr, 'empezando a levantar %s puerto %s' % server_address
sock.bind(server_address)
```

El método accept() acepta una conexión entrante (un cliente) y el método listen() pone al socket en modo servidor.

```
# Escuchando conexiones entrantes
sock.listen(1)

while True:
    # Esperando conexion
    print >>sys.stderr, 'Esperando para conectarse'
    connection, client_address = sock.accept()
```

El método accept() nos devuelve una conexión abierta entre el servidor y el cliente, junto con la dirección del cliente. Los datos de la conexión se leen con el método recv() y se transmiten con el método sendall().

```
while True:
    # Esperando conexion
    print >>sys.stderr, 'Esperando para conectarse'
    connection, client_address = sock.accept()
```

```
try:
    print >>sys.stderr, 'concexion desde', client_address

    # Recibe los datos en trozos y reetransmite
    while True:
        data = connection.recv(19)
        print >>sys.stderr, 'recibido "%s"' % data
        if data:
            print >>sys.stderr, 'enviando mensaje de vuelta al cliente'
            connection.sendall(data)
        else:
            print >>sys.stderr, 'no hay mas datos', client_address
            break

finally:
    # Cerrando conexion
    connection.close()
```

Un adicional al servidor, es la **base de datos** que puede tener asociada, con la cual este es capas de almacenar gran volumen de datos, y procesarlos y así dar un mejor servicio al cliente.

5 Captura de pantalla de Resultados obtenidos

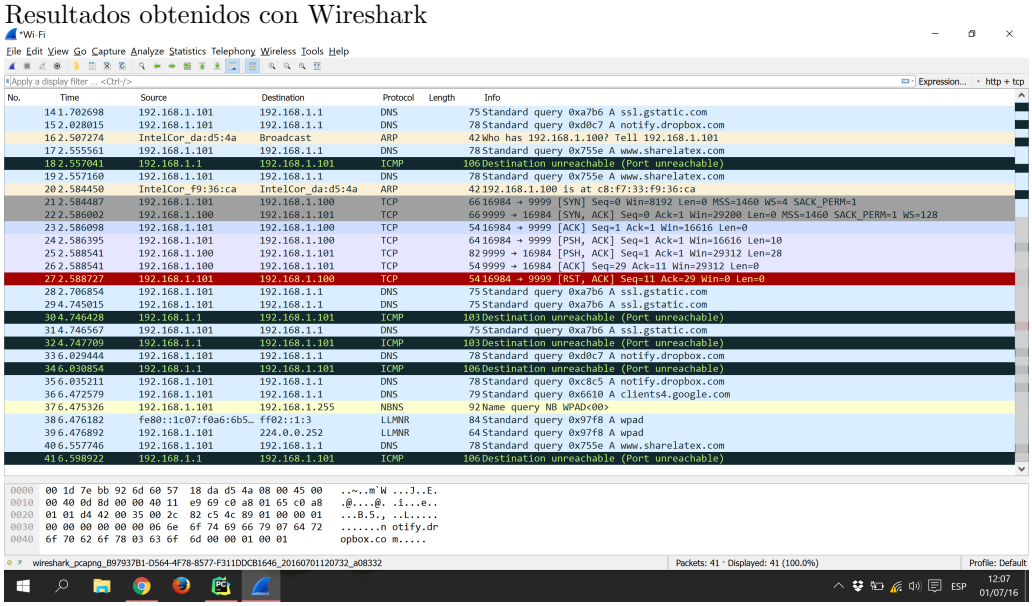
Captura del Cliente:

```
C:\Python27\python.exe C:/Users/Javier/Desktop/Cliente.py
Family  : AF_INET
Type    : SOCK_STREAM
Protocol: IPPROTO_IP

Lo que enviaste fue :  "Hola mundo"
La respuesta del servidor fue "Hola mundo"
Socket cerrado

Process finished with exit code 0
```

Captura del Servidor:

```
C:\Python27\python.exe C:/Users/Javier/Desktop/Servidor.py
starting up on 192.168.0.7 port 9999
waiting for a connection
client connected: ('192.168.0.7', 17585)
received "Hola mundo"
received ""
waiting for a connection
```

Resultados obtenidos con Wireshark



6 Conclusión

El modelo cliente - servidor es un interesante modelo que ha llegado para quedarse. Es lo suficientemente simple y con caracteristicas ventajosas muy importantes. Una invitación a todo aquel lector de este informe es a atreverse a interiorarse e iplementar su propio cliente servidor, dificil no es, y se puede lograr interesantes cosas utilizando su ingenio.

Bibliografía

- Interprocess Communication and Networking
  https://docs.python.org/2/library/ipc.html
- Socket - NetWork Communication https://pymotw.com/2/socket/index.html

- Conceptos básicos de internet, modeo cliente servidor:
  http://www.ite.educacion.es/formacion/materiales/157/cd/m1$_{1}$$_{c}$$onceptos_{b}asicos_{d}e_{i}nternet/modelo_{c}lienteser$