



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: Viernes, 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 través de la tecnología que día a día perfecciona, está el persistente intento de maximizar una comunicación. Es aquí, donde consideramos como una de las ramas por excelencia preocupadas del asunto, **Redes de computadores**.

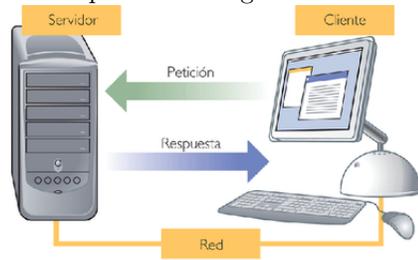
En particular, este informe trata sobre **arquitectura Cliente - Servidor**, el cual es un modelo de una aplicación distribuida en el cual se basa en dos actores: Uno con rol de proveedor de recursos y otro con rol consultor sobre los recursos. Esto será visto en detalle teóricamente y se hará un ejemplo práctico de como se implementa utilizando como lenguaje Python v2.7 utilizando la librería 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 cotidianidad 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 generará.

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 arquitectura 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 líneas principales del código del Servidor. Tener presente que no es el código completo, pues la idea principal es abstraer y con ello simplificar la comprensión 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, 'conexión desde', client_address

    # Recibe los datos en trozos y retransmite
    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 más datos', client_address
            break

finally:
    # Cerrando conexión
    connection.close()

```

Un adicional al servidor, es la **base de datos** que puede tener asociada, con la cual este es capaz 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

The screenshot shows the Wireshark interface with a list of captured packets. The main pane displays a table of packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets include DNS queries, ARP requests, and ICMP destination unreachable messages. The packet list pane at the bottom shows the raw hex and ASCII data for the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
141.782698	192.168.1.101	192.168.1.1	192.168.1.1	DNS	75	Standard query 0xa7b6 A ssl.gstatic.com
152.028015	192.168.1.101	192.168.1.1	192.168.1.1	DNS	78	Standard query 0xd8c7 A notify.dropbox.com
162.587274	IntelCor_da:d5:4a			Broadcast	42	Who has 192.168.1.100? Tell 192.168.1.101
172.555561	192.168.1.101	192.168.1.1	192.168.1.1	DNS	78	Standard query 0x755e A www.sharelatex.com
192.557041	192.168.1.101	192.168.1.101	192.168.1.101	TCP	106	Destination unreachable (Port unreachable)
192.557160	192.168.1.101	192.168.1.1	192.168.1.1	DNS	78	Standard query 0x755e A www.sharelatex.com
202.584450	IntelCor_F9:36:ca	IntelCor_da:d5:4a		ARP	42	192.168.1.100 is at c8:f7:33:f9:36:ca
212.584487	192.168.1.101	192.168.1.100	192.168.1.100	TCP	66	16984 -> 9999 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
222.586002	192.168.1.100	192.168.1.101	192.168.1.101	TCP	66	9999 -> 16984 [SYN, ACK] Seq=0 Ack=1 Win=29280 Len=0 MSS=1460 SACK_PERM=1 WS=128
232.586098	192.168.1.101	192.168.1.100	192.168.1.100	TCP	54	16984 -> 9999 [ACK] Seq=1 Ack=1 Win=16616 Len=0
242.586395	192.168.1.101	192.168.1.100	192.168.1.100	TCP	64	16984 -> 9999 [PSH, ACK] Seq=1 Ack=1 Win=16616 Len=10
252.588541	192.168.1.100	192.168.1.101	192.168.1.101	TCP	82	9999 -> 16984 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=28
262.588541	192.168.1.100	192.168.1.101	192.168.1.101	TCP	54	9999 -> 16984 [ACK] Seq=29 Ack=11 Win=29312 Len=0
272.583727	192.168.1.101	192.168.1.100	192.168.1.100	TCP	54	16984 -> 9999 [RST, ACK] Seq=11 Ack=29 Win=0 Len=0
282.786854	192.168.1.101	192.168.1.1	192.168.1.1	DNS	75	Standard query 0xa7b6 A ssl.gstatic.com
294.745015	192.168.1.101	192.168.1.1	192.168.1.1	DNS	75	Standard query 0xa7b6 A ssl.gstatic.com
304.746428	192.168.1.1	192.168.1.101	192.168.1.101	ICMP	103	Destination unreachable (Port unreachable)
314.746567	192.168.1.101	192.168.1.1	192.168.1.1	DNS	75	Standard query 0xa7b6 A ssl.gstatic.com
324.747709	192.168.1.101	192.168.1.101	192.168.1.101	ICMP	103	Destination unreachable (Port unreachable)
336.029444	192.168.1.101	192.168.1.1	192.168.1.1	DNS	78	Standard query 0xd8c7 A notify.dropbox.com
346.030854	192.168.1.1	192.168.1.101	192.168.1.101	ICMP	106	Destination unreachable (Port unreachable)
356.035211	192.168.1.101	192.168.1.1	192.168.1.1	DNS	78	Standard query 0xc8c5 A notify.dropbox.com
366.472579	192.168.1.101	192.168.1.1	192.168.1.1	DNS	79	Standard query 0xc610 A clients4.google.com
376.475326	192.168.1.101	192.168.1.255	192.168.1.255	MIBS	92	Name query NB WPAD:000
386.476182	f680:1107:f0a6:605::f602:1113	224.0.0.252	224.0.0.252	LLMNR	84	Standard query 0x97f8 A wpad
396.476892	192.168.1.101	192.168.1.1	192.168.1.1	DNS	64	Standard query 0x97f8 A wpad
406.557746	192.168.1.101	192.168.1.1	192.168.1.1	DNS	78	Standard query 0x755e A www.sharelatex.com
416.598922	192.168.1.1	192.168.1.101	192.168.1.101	ICMP	106	Destination unreachable (Port unreachable)

```
0000 00 1d 7e bb 92 6d 60 57 18 da d5 4a 08 00 45 00  ...mW ...E.
0010 00 40 0d 00 00 00 40 11 e9 69 c0 a8 01 05 c0 a8  @...@.i...e.
0020 01 01 04 42 00 35 00 2c 62 c5 4c 89 01 00 00 01  ...B.S...l.....
0030 00 00 00 00 00 06 0e 6f 74 69 66 79 07 64 72  ....n otify.dr
0040 6f 70 62 6f 78 03 63 6f 6d 00 00 01 00 01      opbox.co m.....
```

6 Conclusión

El modelo cliente - servidor es un interesante modelo que ha llegado para quedarse. Es lo suficientemente simple y con características ventajosas muy importantes. Una invitación a todo aquel lector de este informe es a atreverse a interiorarse e implementar su propio cliente servidor, difícil 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, modo cliente servidor:
http://www.ite.educacion.es/formacion/materiales/157/cd/m11c_onceptos_basicos_de_internet/modelo_cliente_servidor/