Programación de Sockets

Intergantes: Bryan Jeldes, Eduardo Arancibia, Fabricio Rosales, Mathias Oyarzun

Universidad Técnica Federico Santa María, 28-07-2014

Resumen

En la década de los 80's, cuando se utilizaba la red ARPANET y ARPA Internet, el sistema operativo modificado [1]BSD UNIX, necesitaba facilitar el diseño de aplicaciones que utilizaran dichas redes; por tal motivo, se creó la Interfaz de Programación de Sockets. [2]Sockets, es un método de comunicación programable por el cual un proceso puede conectarse y enviar flujos de datos a otros procesos que están dentro de un mismo sistema, o en una maquina completamente distinta.

En nuestro proyecto nos centramos netamente en programar sockets que conecte procesos en máquinas distintas. Mostramos la programación de sockets en algunos lenguajes, relacionamos funciones incluidas en la programación, conocimos que necesitaba un socket para que funcionara, etc. Y en la práctica, tratamos de entablar la comunicación entre un programa cliente y un programa servidor, que estaba alojado en una Raspberry PI, a través del lenguaje de programación Python, e investigamos con el programa Wireshark como los programas enviaban la información con sus respectivos paquetes.

Finalmente, pudimos hacer efectiva la comunicación a la Raspberry P; enviábamos mensajes del cliente, ejecutado en un notebook, al servidor y comprobamos con Wireshark los paquetes enviados.

[1] Programación de Sockets por Xavier Perramon Tornil y Enric Peig Olivé, UOC.

[2] http://es.wikipedia.org/wiki/Socket_de_Internet, [1]

Introducción

La presente investigación se refiere al uso de los sockets, y en específico a los sockets que poseen su espacio de nombre en formato Internet, es decir, procesos que se comunican alojados en máquinas distintas.

El objetivo principal era averiguar cómo se programaba uno de estos conectores, hacerlo funcionar en la práctica y estudiar los componentes que se necesitaban para que este se ejecutara. Con este fin, se planteó la idea de hacer un programa cliente y un programa servidor que intercambiaran mensajes entre sí, como una especie de chat arcaico; el cliente seria alojado en un notebook, con un código específico, y el servidor en una Raspberry PI.

En pro del objetivo principal, se usó el lenguaje de programación Python para llevar a cabo estos programas (para fines prácticos, Python era la mejor opción, ya que era simple y rápido para mostrar el funcionamiento del codigo), se buscaron los componentes necesarios para que un socket funcionara, se programó un código en Python que enviara mensajes a otro programa ejecutado en otra máquina, en este caso una Raspberry PI.

En general, se tratara de evaluar la complejidad de programar sockets, estudiar sus componentes y funciones, y además, comprobar los paquetes que este envía al servidor y viceversa.

Programación de Sockets

En primera instancia, para programar un socket, escogimos una especie de cliente-servidor(modelo P2P) que se comunicaran a través de mensajes. Ahora se mostraran las funciones utilizadas para hacer funcionar los códigos.

El código del programa Cliente:

```
from socket import *
HOST = '192.168.1.107'
PORT = 6975
s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT))
while True:
    message = raw_input("Tu Mensaje: ")
    s.send(message)
    print "Esperando Respuesta"
    reply = s.recv(1024)
    print "Recibido ", repr(reply)
s.close()
```

-HOST y PORT: Se inicializan estas variables con valores fijos, ya que para fines prácticos, lo conectaremos al servidor con un IP y puerto prefijado (dentro de un red doméstica).

-socket(AF_INET, SOCK_STREAM): función que crea el socket. AF_INET hace alusión a la comunicación en máquinas distintas. SOCK_STREAM indica que se transportaran los datos vía TCP .

-s.connect(HOST, PORT): conecta al servidor prefijado.

-s.send(message): envía el mensaje escrito por el cliente.

-s.recv(): recibe un mensaje proveniente del servidor.

-s.close(): Cierra la conexión.

El código del programa Servidor:

```
from socket import *

HOST = ''
PORT = 6975
s = socket(AF_INET, SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Conectado por', addr
while True:
    data = conn.recv(1024)
    print "Recibido", repr(data)
    reply = raw_input("respuesta: ")
    conn.sendall(reply)

conn.close()
```

-socket(AF_INET, SOCK_STREAM): Lo mismo que en el cliente, AF_INET para conectarse a otra maquina, y SOCK_STREAM para usar transporte via TCP.

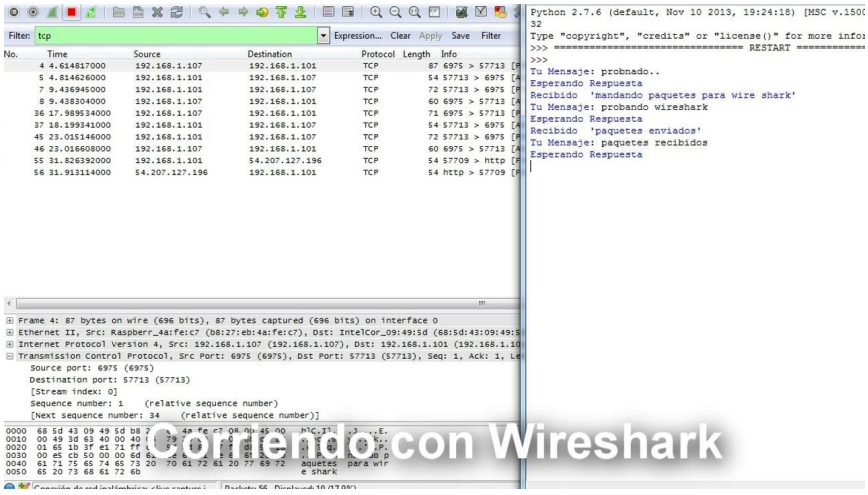
-s.bind(HOST, PORT): Se le asigna una dirección y un puerto al servidor. Al igual que antes estos ya están prefijados, sin embargo, HOST no es necesario en este caso, solo el puerto.

s.listen(1): Indica cuantas conexiones aceptara. En este caso aceptaría una sola conexión.

s.accept(): Acepta la conexión entrante. Esta variable es una tupla y es igualada a las variables conn y addr.

Conn.recv(1024),Conn.sendall(reply), con.close(): .recv recibe lo que le envio el cliente, .sendall(reply) envía a todos lo que esta en reply y con.close() cierra la conexión.

Finalmente, se puede comprobar con WIRESHARK que al enviar mensajes al servidor, estos se comunican gracias a los sockets que utilizan los puertos respectivos:57713 para el cliente y 6975 para el servidor.



Conclusiones

En este proyecto se averiguo el cómo se programaba un socket en bruto, usando algún lenguaje de programación, que en este caso fue [3]Python. Se investigaron las funciones que utilizaban los códigos para entender el cómo funcionaba un socket, como se comunicaba con un proceso alojado en otra máquina. Se programó el cliente en un notebook y el servidor en una Raspberry PI, y se comprobó que se pueden comunicar a través de sockets, verificándolo con Wireshark. [4]Además en el proceso de investigación se observo cómo se programaba en otros lenguajes que no eran python como C y Java, estos dos ultimos fueron abordados de manera general sin embargo se llegó al concenso de que el sistema es igual para todos, lo que cambia es el cómo se programa pero en esencia todos usan las mismas funciones al fin y al cabo.

En conclusión, los sockets cumplieron ampliamente la tarea que se les dio cuando fueron creados para ARPA Internet, y creemos que serán mejorados en el futuro facilitando aún más el diseño de programas que necesiten comunicarse con sistemas.

[3] http://victorpando.blogspot.com/2008/12/programacin-de-sockets-con-python.html

[4] http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html y Programación de Sockets por Xavier Perramon Tornil y Enric Peig Olivé, UOC.

Referencias

- http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html
- http://es.wikipedia.org/wiki/Socket_de_Internet
- http://victorpando.blogspot.com/2008/12/programacin-de-sockets-con-python.html
- Programación de Sockets por Xavier Perramon Tornil y Enric Peig Olivé, UOC.