Sistemas Distribuidos
Grado en Ingeniería Informática

Jose Alberto Jiménez Serrano
Adrián Garrido Adriano

1. Introducción

El objetivo de este trabajo es la implementación de un sistema de copias de seguridad de archivos implementada en python y haciendo uso de ZeroMQ.

2. Descripción del problema a solucionar

Queremos disponer de un sistema de copias de seguridad, mediante una red local, que nos garantice la seguridad de nuestros archivos. Para ello, la copia de dichos datos se almacenará en un servidor central. Haciendo uso de Python y de las características que nos ofrece ZeroMQ, hemos realizado este sistema de fácil utilización.

Supongamos una red de nodos cuya información es de vital importancia y no se permite ningún tipo de pérdida de información. Este sistema permite, como ya hemos dicho, crear copias de seguridad de los datos y almacenarlos en un servidor central, asegurándonos que no haya pérdidas de los mismos.

2.1. Descripción del sistema

Hemos implementado dicho sistema utilizando Python y todas las características y bibliotecas que nos ofrece ZeroMQ.

El servidor central es el encargado de planificar las copias de seguridad y y también de hacer las peticiones a los diferentes nodos que componen la red. Para ello, este servidor central pasará por todos los nodos de la red cada cierto tiempo comprobando si alguno de estos nodos está activo. Si un nodo se encuentra activo se inicia la copia de los datos desde el nodo hasta el servidor central (en este servidor se almacenarán todas las copias de los datos).

La comunicacción entre el servidor central y los nodos se lleva a cabo de la siguiente forma:

1. El servidor central envía peticiones a los nodos según la dirección IP.
2. El nodo le responde con el archivo sobre el que se realizará la copia.

3. El servidor recibirá el archivo y lo guardará en la carpeta correspondiente al nodo origen.

4. Este proceso se repite para cada nodo de la red.

3. Implementación

Para la implementación creamos un archivo client.py, para los nodos, y otro server.py, para el servidor principal.

- Cliente:

```
DIR = 'files/bd'

def enviarFichero(fichero, sock):
    '''
    Función que envía un fichero, que es argumento de entrada.
    '''
    fn = open(fichero, 'rb')
    stream = True

    # Empieza la lectura del archivo
    while stream:
        stream = fn.read(128)
        if stream:
            sock.send(stream, zmq.SNDMORE)
        else:
            sock.send(stream)

    print "Archivo " + fichero + " enviado"


def server():
    '''
    Servidor. En este caso actúa como cliente. Espera la petición del servidor
    y envía el archivo correspondiente.
    '''

    context = zmq.Context(1)
    sock = context.socket(zmq.REP)
    sock.bind('tcp://*:4545')

    print "Servidor cliente en ejecucion"
    # Bucle principal
    while True:
        # Obtenemos el mensaje
        msg = sock.recv()
        # Concatenamos el directorio que se est
        if msg == 'copy':
            # Comprobamos que el fichero existe
            if not os.path.isfile(DIR):
                sock.send('')
                print "El archivo " + DIR + " no se encuentra en el sistema"
                continue
            enviarFichero(DIR, sock)
        else:
            print "Opción incorrecta"

if __name__ == '__main__':
    server()
```

Figura 1: Código del cliente

- Servidor:

```
import zmq
import os
import time
import sys
import subprocess

rango = [9, 12]

def main():
    while True:
        time.sleep(5)
        print "Nuevo evento"
        for ping in rango:
            time.sleep(2)
            address = "127.0.0" + str(ping)
            res = subprocess.call(['ping', '-c', '1', '-W', '1', address], stdout=open(os.devnull, "wb"), stderr=open(os.devnull, "wb"))
            if res == 0:
                print "Ping correcto. ", address
                context = zmq.Context()
                socket = context.socket(zmq.REQ)
                socket.connect('tcp://'+address+':4545')
                dest = open(os.path.basename('files/bd'), 'w+')
                socket.send('copy')

                while True:
                    # start grabing data
                    data = socket.recv()
                    # print data
                    # write the chunk to the file
                    dest.write(data)
                    if not socket.getsockopt(zmq.RCVMORE):
                        # If there is not more data to send, then break
                        break

                dest.close()
                print "Copia de seguridad realizada"
                time.sleep(0.5)
                os.chdir('..')


if __name__ == '__main__':
    #get_file(sys.argv[1])
    main()
```

Figura 2: Código del servidor

4. Pruebas

Antes de ejecutar la aplicación, hay que configurar el cliente y el servidor. En el cliente hay que indicar el directorio de monitorización, y en el servidor debemos indicar el tiempo entre cada iteración, la IP de la red en la que se trabaja y por último el rango de nodos que se van a utilizar.

Para ejecutarlo basta con dos comandos: client.py y server.py

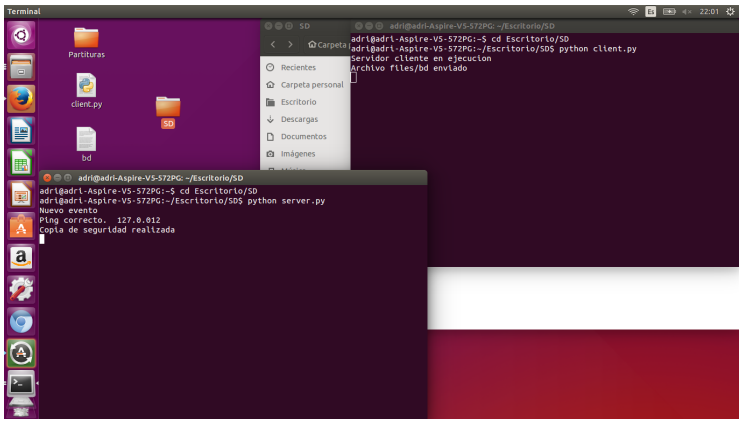

Figura 3: Ejecución