Technical Problem Share [15/04/2020]

Edit by Liu Yuancheng

1. Problem

Type: Python network communication.
Description: ICMP message caused the echo-UDP server raise exception.

2. Problem Back Ground

We created a UDP data update client and the echo data handling server lib module by python-3.7.4. The server will call the plugged in method to auto response the client's request if the client's <sendMsg()> response flag was set to True

This is the client and server module program:
Git hub link: https://github.com/LiuYuancheng/ProgramTools/blob/master/socketComm/src/udpCom.py

```
#!/usr/bin/python
#-----------------------------------------------------------------------------
# Name:        udpCom.py
#
# Purpose:     This module will provide a UDP client and server communication API.
#
# Author:      Yuancheng Liu
#
# Created:     2019/01/15
# Copyright:
# License:
#-----------------------------------------------------------------------------
import time
import socket

BUFFER_SZ = 4096    # UDP buffer size.
RESP_TIME = 0.01

#-----------------------------------------------------------------------------
#-----------------------------------------------------------------------------
class udpClient(object):
    """ UDP client module."""
    def __init__(self, ipAddr):
        """ Create an ipv4 (AF_INET) socket object using the udp protocol (SOCK_DGRAM)
            init example: client = udpClient(('127.0.0.1', 502))
        """
        self.ipAddr = ipAddr
        self.client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

#--udpClient-----------------------------------------------------------------
    def sendMsg(self, msg, resp=False, ipAddr=None):
        """ Convert the msg to bytes and send it to UDP server.
            - resp: server response flag, method will wait server's response and
                return the bytes format response if it is set to True.
        """
        if not ipAddr is None: self.ipAddr = ipAddr  # reset ip address.
        if self.client is None: return None # Check whether disconnected.
        if not isinstance(msg, bytes): msg = str(msg).encode('utf-8')
        self.client.sendto(msg, self.ipAddr)
        if resp:
            try:
                self.client.settimeout(20)
                data, _ = self.client.recvfrom(BUFFER_SZ)
                return data
            except ConnectionResetError as error:
                print("udpClient: Can not connect to the server!")
                print(error)
                self.disconnect()
                return None
        return None

#--udpClient-----------------------------------------------------------------
    def disconnect(self):
        """ Send a empty logout message and close the socket."""
        self.sendMsg('123', resp=False)
        time.sleep(RESP_TIME)# sleep very short while before close the socket to \
        # make sure the server have enought time to handle the close method, when \
        # server computer is fast, this is not a problem.

        # Call shut down before close: https://docs.python.org/3/library/socket.html#socket
.socket.shutdown
        self.client.shutdown(socket.SHUT_RDWR)
        self.client.close()
        self.client = None
#----------------------------------------------------------------------------
#----------------------------------------------------------------------------
class udpServer(object):
    """ UDP server module."""
    def __init__(self, parent, port):
        """ Create an ipv4 (AF_INET) socket object using the tcp protocol (SOCK_STREAM)
            init example: server = udpServer(None, 5005)
        """
        self.server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.server.bind(('0.0.0.0', port))
        self.terminate = False  # Server terminate flag.

#--udpServer---------------------------------------------------------------
    def serverStart(self, handler=None):
        """ Start the UDP server to handle the incomming message."""
        while not self.terminate:
            data, address = self.server.recvfrom(BUFFER_SZ)
            print("Accepted connection from %s" % str(address))
            msg = handler(data) if not handler is None else data
            if not msg is None:  # don't response client if the handler feed back is None
                if not isinstance(msg, bytes): msg = str(msg).encode('utf-8')
                self.server.sendto(msg, address)
        self.server.close()

#--udpServer---------------------------------------------------------------
    def serverStop(self):
        self.terminate = True

#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
# Test case program: udpComTest.py
```

This is the Library module test-Case program:
Github link:
https://github.com/LiuYuancheng/ProgramTools/blob/master/socketComm/src/udpComTest.py

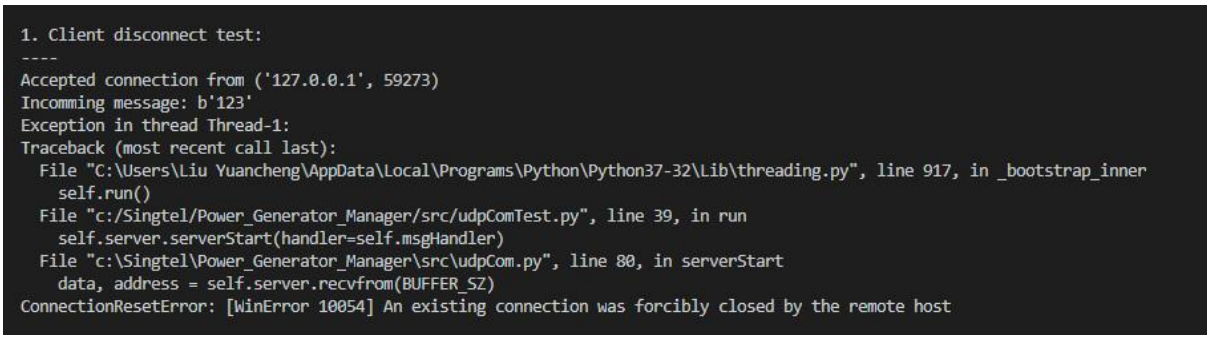
```
#!/usr/bin/python
#-------------------------------------------------------------------------------
# Name:        udpComTest.py
#
# Purpose:     This module will provide a muti-thread test case program to test
#              the UDP communication modules by using port 5005.
#
# Author:      Yuancheng Liu
#
# Created:     2019/01/15
# Copyright:
# License:
#-------------------------------------------------------------------------------

import time
```

```python
import threading    # create multi-thread test case.
import udpCom

UDP_PORT = 5005
#-----------------------------------------------------------------------------
#-----------------------------------------------------------------------------
class testThread(threading.Thread):
    """ Thread to test the UDP server/insert the tcp server in other program."""
    def __init__(self, parent, threadID, name):
        threading.Thread.__init__(self)
        self.threadName = name
        self.server = udpCom.udpServer(None, UDP_PORT)

    def msgHandler(self, msg):
        """ The test handler method passed into the UDP server to handle the
            incoming messages.
        """
        print("Incomming message: %s" %str(msg))
        return msg

    def run(self):
        """ Start the udp server's main message handling loop."""
        print("Server thread run() start.")
        self.server.serverStart(handler=self.msgHandler)
        print("Server thread run() end.")
        self.threadName = None # set the thread name to None when finished.

    def stop(self):
        """ Stop the udp server. Create a endclient to bypass the revFrom() block."""
        self.server.serverStop()
        endClient = udpCom.udpClient(('127.0.0.1', UDP_PORT))
        endClient.disconnect()
        endClient = None

#-----------------------------------------------------------------------------
#-----------------------------------------------------------------------------
def testCase(mode):
    print("Start UDP client-server test. test mode: %s \n" % str(mode))
    tCount, tPass = 0, True  # test fail count and test pass flag.
    if mode == 1:
        print("Start the UDP Server.")
        servThread = testThread(None, 0, "server thread")
        servThread.start()
        print("Start the UDP Client.")
        client = udpCom.udpClient(('127.0.0.1', UDP_PORT))
        # test case 0
        print("0. HearBeat message test:\n----")
        tPass = True
        for i in range(3):
            msg = "Test data %s" %str(i)
            result = client.sendMsg(msg, resp=True)
            tPass = tPass and (msg.encode('utf-8') == result)
        if tPass: tCount += 1
        print("Test passed: %s \n----\n" % str(tPass))

        # test case 1
        print("1. Client disconnect test:\n----")
        tPass = True
        client.disconnect()
        client = None
        try:
            client.sendMsg('Testdata', resp=True)
            tPass = False
        except:
            tPass = True
        if tPass: tCount += 1
        print("Test passed: %s \n----\n" % str(tPass))

        # test case 2
        print("2. Server stop test:\n----")
        servThread.stop()

        time.sleep(1)  # wait 1 second for all the sorcket close.
        tPass = (servThread.threadName is None)
        if tPass: tCount += 1
        print("Test passed: %s \n----\n" % str(tPass))

        # test case 3
        print("3. Client timeout test:\n----")
        client = udpCom.udpClient(('127.0.0.1', UDP_PORT))
        resp = client.sendMsg('Testdata', resp=True)
        tPass = (resp is None)
        if tPass: tCount += 1
        print("Test passed: %s \n----\n" % str(tPass))

        print(" => All test finished: %s/4" % str(tCount))
        client = servThread = None
    else:
        print("Add more other exception test here.")
#----------------------------------------------------------------------------
if __name__ == '__main__':
    testCase(1)
```

In out UDP client program, when we want to disconnect from the server, we will send a disconnect message first (to make it simple to track currently we use str ‘123’), then we shutdown and close the socket. But if we don’t sleep a shot while(0.001Sec) as below:

```
    def disconnect(self):
        """ Send a empty logout message and close the socket."""
        self.sendMsg('123', resp=False)
        time.sleep(RESP_TIME) <= Without this line, there will be a error !
        self.client.shutdown(socket.SHUT_RDWR)
        self.client.close()
        self.client = None
```

The server’s recvFrom() method will report a **ConnectionResetError**:



```
1. Client disconnect test:
----
Accepted connection from ('127.0.0.1', 59273)
Incomming message: b'123'
Exception in thread Thread-1:
Traceback (most recent call last):
  File "C:\Users\Liu Yuancheng\AppData\Local\Programs\Python\Python37-32\Lib\threading.py", line 917, in _bootstrap_inner
    self.run()
  File "c:/Singtel/Power_Generator_Manager/src/udpComTest.py", line 39, in run
    self.server.serverStart(handler=self.msgHandler)
  File "c:\Singtel\Power_Generator_Manager\src\udpCom.py", line 80, in serverStart
    data, address = self.server.recvfrom(BUFFER_SZ)
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host
```

As the socket has be closed, logically there should not anything send from the client to the server, and UDP is “one way” communication.

3. Problem Test Case Program

Copy <udpCom.py> and <udpComtest.py> in same folder and run:

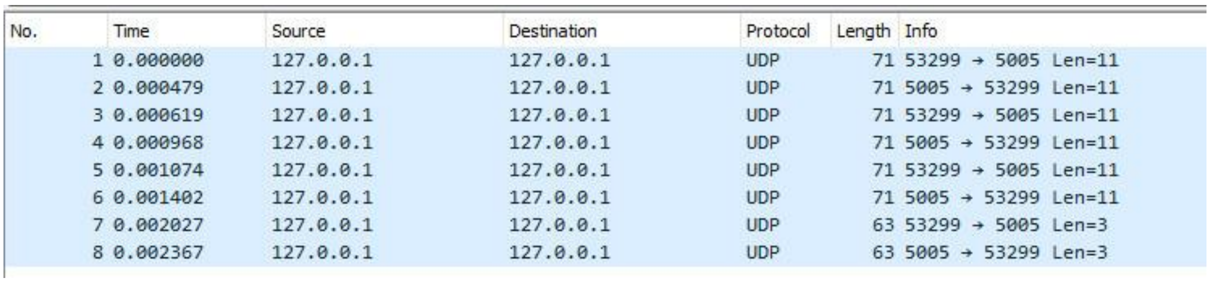
```
$ python udpComtest.py
```

4. Problem Analysis

Based on the exception message, we guess there must be some unexpected message got to the server side. So we try to use wireShark to capture the traffic:

⇨ Normal Case:

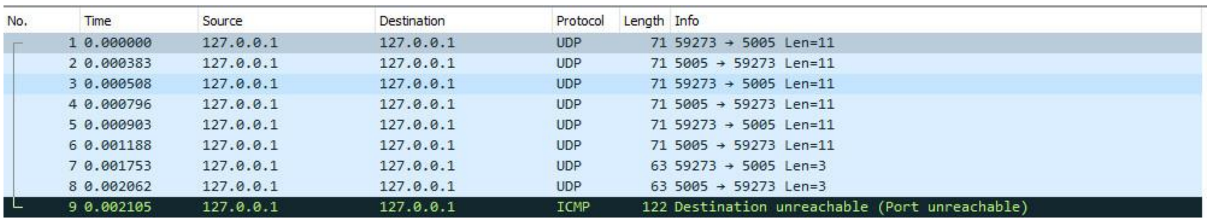
This is the wireShark captured network communication between client and server when we call sleep() to make the program wait a short while (no exception):



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	71	53299 → 5005 Len=11
2	0.000479	127.0.0.1	127.0.0.1	UDP	71	5005 → 53299 Len=11
3	0.000619	127.0.0.1	127.0.0.1	UDP	71	53299 → 5005 Len=11
4	0.000968	127.0.0.1	127.0.0.1	UDP	71	5005 → 53299 Len=11
5	0.001074	127.0.0.1	127.0.0.1	UDP	71	53299 → 5005 Len=11
6	0.001402	127.0.0.1	127.0.0.1	UDP	71	5005 → 53299 Len=11
7	0.002027	127.0.0.1	127.0.0.1	UDP	63	53299 → 5005 Len=3
8	0.002367	127.0.0.1	127.0.0.1	UDP	63	5005 → 53299 Len=3

⇨ Exception Case:

This is the captured network communication between client and server when we don't call sleep ():



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	71	59273 → 5005 Len=11
2	0.000383	127.0.0.1	127.0.0.1	UDP	71	5005 → 59273 Len=11
3	0.000508	127.0.0.1	127.0.0.1	UDP	71	59273 → 5005 Len=11
4	0.000796	127.0.0.1	127.0.0.1	UDP	71	5005 → 59273 Len=11
5	0.000903	127.0.0.1	127.0.0.1	UDP	71	59273 → 5005 Len=11
6	0.001188	127.0.0.1	127.0.0.1	UDP	71	5005 → 59273 Len=11
7	0.001753	127.0.0.1	127.0.0.1	UDP	63	59273 → 5005 Len=3
8	0.002062	127.0.0.1	127.0.0.1	UDP	63	5005 → 59273 Len=3
9	0.002105	127.0.0.1	127.0.0.1	ICMP	122	Destination unreachable (Port unreachable)

We can see there is an additional ICMP protocol message to identify the destination unreachable (marked as black), ICMP (Internet Control Message Protocol) is an error-reporting protocol network devices like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets.

This kind of problem happens when we send a message to the server and close the socket as some as possible. Because the system haven't finished cleaning the socket buffer and we already started shutdown and close the socket, the system will send a ICMP message to the server side. That caused the server raised the exception. If we switched to a high performance computer, that problem can be avoid even we don't call the <time.sleep()>

5. **Problem Situation Usage in Cyber Security**

So when we design a UDP echo server program, we need to consider about this kind of situation, the attacker can captured this ICMP message and repeat sending it to the server to try to hang the server program.