How to communicate in between 2 different instances of VC Premium in 2 different machine via TCP/IP (or UDP) socket.

Jamal Muhammad

jamal.muhammad@visualcomponents.com

Contents

Python TCP/IP network programming

Python provides two levels of access to network services. At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

This chapter gives you understanding on most famous concept in Networking - Socket Programming.

What is Sockets?

Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

Sockets have their own vocabulary:

Term	Description
domain	The family of protocols that is used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.
type	The type of communications between the two endpoints, typically SOCK_STREAM for connection-oriented protocols and SOCK_DGRAM for connectionless protocols.
protocol	Typically zero, this may be used to identify a variant of a protocol within a domain and type.
hostname	The identifier of a network interface: ▪ A string, which can be a host name, a dotted-quad address, or an IPV6 address in colon (and possibly dot) notation ▪ A string "<broadcast>", which specifies an INADDR_BROADCAST address. ▪ A zero-length string, which specifies INADDR_ANY, or ▪ An Integer, interpreted as a binary address in host byte order.
port	Each server listens for clients calling on one or more ports. A port may be a Fixnum port number, a string containing a port number, or the name of a service.

The socket module

To create a socket, you must use the socket.socket() function available in socket module, which has the general syntax −

s = socket.socket (socket_family, socket_type, protocol=0)

Here is the description of the parameters −

socket_family: This is either AF_UNIX or AF_INET, as explained earlier.

socket_type: This is either SOCK_STREAM or SOCK_DGRAM.

protocol: This is usually left out, defaulting to 0.

Once you have socket object, then you can use required functions to create your client or server program. Following is the list of functions required –

Server socket methods

Method	Description
s.bind()	This method binds address (hostname, port number pair) to socket.
s.listen()	This method sets up and start TCP listener.
s.accept()	This passively accept TCP client connection, waiting until connection arrives (blocking).

Client socket methods

Method	Description
s.connect()	This method actively initiates TCP server connection.

General socket methods

Method	Description
s.recv()	This method receives TCP message
s.send()	This method transmits TCP message
s.recvfrom()	This method receives UDP message
s.sendto()	This method transmits UDP message
s.close()	This method closes socket
socket.gethostname()	Returns the hostname.

Reference material to read more about python TCP/IP socket connection -

https://docs.python.org/3.0/library/socket.html

https://pymotw.com/2/socket/tcp.html

First example with only python IDLE

Simple Client side program

Try to first practice the basic of socket connection by doing the following socket programming in Python IDLE (2.7.1 was the version used here)

client_TCP_IP.py - C:\Users\muhamJa1\Downloads\client_TCP_IP.py

File Edit Format Run Options Windows Help

```
import sys
import time
from socket import socket, AF_INET, SOCK_DGRAM

# AF_INET - this constant represent the address(and protocol) family,used for the first argument to socket().
# SOCK_DGRAM - this constant represent the socket types, used for the second argument to socket().
#              (Only SOCK_STREAM and SOCK_DGRAM appear to be generally useful.)

SERVER_IP   = '192.168.8.129'
PORT_NUMBER = 4006
SIZE = 1024
print ("Test client sending packets to IP {0}, via port {1}\n".format(SERVER_IP, PORT_NUMBER))

mySocket = socket( AF_INET, SOCK_DGRAM )

i = 0
while i < 10:
    mySocket.sendto(("[%f,%f,%f,%f,%f,%f]"%(i,i,i,i,i,i)).encode('utf-8'),(SERVER_IP,PORT_NUMBER))
    #time.sleep(0.02)
    time.sleep(1)
    i = i + 1


sys.exit()
```

Simple Server side program

The following server side is running in the Virtual Machine. In this example VM Ware.

server_2.py - C:\Users\Visual Components\Downloads\server_2.py

File Edit Format Run Options Windows Help

```
from socket import *
import sys

#hostName = "localhost"
#hostName = gethostbyname( '0.0.0.0' )
hostName = gethostbyname(gethostname())
PORT_NUMBER = 4006
SIZE = 1024

mySocket = socket( AF_INET, SOCK_DGRAM )
mySocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
mySocket.bind( (hostName, PORT_NUMBER) )

print ("Test server receiving packets from local host IP {0}, via port {1}\n".format(hostName, PORT_NUMBER))

while True:
    (data,addr) = mySocket.recvfrom(SIZE)
    print data

sys.exit()
```

To make sure that user can ping into virtual machine from host pc and vice versa you can try to turn off Windows firewall in virtual machine and then ping both ways using [ping IP address] from command prompt.

TCP/IP socket connection in between two VC programs in 2 different machines

In this example an instance of VC Premium is going to run in the host machine and another instance in a virtual machine i.e. VM Ware in this case. The host pc will act as a client and the guest machine will act as server. Our goal is pretty simple. To move a robot in VC Premium in the host pc and emulate the same movement in VC Premium running in guest machine.

There are basically 2 main components in this example.

TCP_IP Client

TCP_IP Server

They are attached with this instruction package. How it works:

 # TCP_IP Client

- Establishes socket connection in the OnStart() event.
- Gets a handle to the robot controller and iterates through the joint values of the robot.
- These joint values are then transferred via socket to the sever side i.e. VC Premium running in virtual machine.
- An important concept is to use non-blocking socket connection. Our VC software is single thread so using blocking socket connection will stop the simulation from running.

 # TCP_IP Server

- Establishes a socket connection to read from its localhost and the port set by user, in this case port number 4006.
- Gets a handle to robot controller.
- Waits for incoming message (socket data) and then populates a list which contains all the joint values for each packet data.
- Robot joints are moved to those values via iteration.
- An important concept is to use non-blocking socket connection. Our VC software is single thread so using blocking socket connection will stop the simulation from running.

All the necessary #components #python script are attached here by with the instruction folder.

TCP_IP Client

```
from vcScript import *
from socket import socket, AF_INET, SOCK_DGRAM, SOCK_STREAM

# AF_INET - this constant represent the address(and protocol) family,used for the first argument to socket().
# SOCK_DGRAM - this constant represent the socket types, used for the second argument to socket().
#              (Only SOCK_STREAM and SOCK_DGRAM appear to be generally useful.)

SERVER_IP   = '192.168.8.129'
PORT_NUMBER = 4006
SIZE = 1024

def OnStart():
  print ("Test client sending packets to IP {0}, via port {1}\n".format(SERVER_IP, PORT_NUMBER))
  ServerTCP_IP()

def ServerTCP_IP():
  global mySocket

  mySocket = socket( AF_INET, SOCK_DGRAM )
  mySocket.setblocking(0)

def OnRun():
  global mySocket

  delay(0.1)
  while True:
    controller = robot.findBehavioursByType(VC_ROBOTCONTROLLER)[0]
    j1 = controller.getJointValue(0)
    j2 = controller.getJointValue(1)
    j3 = controller.getJointValue(2)
    j4 = controller.getJointValue(3)
    j5 = controller.getJointValue(4)
    j6 = controller.getJointValue(5)
    #mySocket.sendto(("[%f,%f,%f,%f,%f,%f]"%(j1,j2,j3,j4,j5,j6)).encode('utf-8'),(SERVER_IP,PORT_NUMBER))
    mySocket.sendto(("[%f,%f,%f,%f,%f,%f]"%(j1,j2,j3,j4,j5,j6)),(SERVER_IP,PORT_NUMBER))
    delay(0.001)
  sim.halt()

sim = getSimulation()
app = getApplication()
comp = getComponent()
RobotName = comp.getProperty('RobotName').Value
robot = app.findComponent(RobotName)
```

TCP_IP Server

```
from vcScript import *
from socket import *

hostName = gethostbyname(gethostname())
PORT_NUMBER = 4006
SIZE = 1024

def OnStart():
  print ("Test server receiving packets from local host IP {0}, via port {1}\n".format(hostName, PORT_NUMBER))
  establishSocket()

def establishSocket():
  global mySocket

  mySocket = socket( AF_INET, SOCK_DGRAM )
  mySocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
  mySocket.bind( (hostName, PORT_NUMBER) )
  mySocket.setblocking(0)

def OnRun():
  global mySocket

  delay(0.001)
  rc = robot.findBehavioursByType(VC_ROBOTCONTROLLER)[0]
  mt = rc.createTarget()
  mt.MotionType = VC_MOTIONTARGET_MT_JOINT
  mt.UseJoints = True
  jv = mt.JointValues

  while True:
    try:
      (data,addr) = mySocket.recvfrom(SIZE)
      data = data.translate(None, '[]')
      data1 = data.split(',')
      for i in range(len(data1)):
        val = float(data1[i])
        jv[i] = float(val)
      mt.JointValues = jv
      controller.moveImmediate(mt)
    except error:
      delay(0.001)

app = getApplication()
comp = getComponent()
RobotName = comp.getProperty('RobotName').Value
robot = app.findComponent(RobotName)
controller = robot.findBehavioursByType(VC_ROBOTCONTROLLER)[0]
```

How to make this example work

- Load the component [TCP_IP Client.vcmx] into VC Premium and then load any 6-axis robot from eCat e.g Kawasaki CX165L.
- Teach the robot some points and then try to make the Cycle time of movement in between points a big more time e.g. 2 or 3 seconds.
- Now there is property in the component [TCP_IP Client.vcmx] called (RobotName). In here copy paste the robot's name which you want to get Joint values of i.e. CX165L in this case.
- Run VM Ware and run VC Premium in there as well.
- Load the component [TCP_IP Server.vcmx] into 3D world and then load a CX165L into the same layout. In the component property TCP_IP Server > RobotName now write the name of the robot in the layout i.e. CX165L in this case.
- Ping in between host pc and guest machine(VM ware) to make sure that information is passed in between them without any blockage. In case blockage please turn OFF the Firewall of the virtual machine.
- Now first run the simulation in the virtual machine and then run the simulation in the host pc. You will see data packets are send from host pc to guest machine via socket and the robot in VC Premium is emulating the movement of the robot running in host pc > VC Premium.