---

Title: NETWORK END-TO-END HEALTH CHECKER

Date: 2017-12-11

Author: Tyler

---

I am helping a friend with a business he is starting. One thing he wants is the ability to test a network health remotely and regularly from an endpoint, not just using the network providers tools.

Step one of this project was writing a server where we could store all this information. I might break down that project, step by step, but for now, you can see it [here](https://github.com/connectivtech/dataserver). This is my first Django project so if you have feedback or advice please reach out.

Step 2, and what I will be covering in this article is building the software that goes on a Raspberry Pi to ping the network and send that ping to the server.

The first thing we want to do is create a little script we will attach to a cron job. We start with the shebang, or the hashbang. I read this is important so the bash knows what interpreter to use... I just know if I don't have this, things fall apart.

#! /usr/bin/env python

next, we need to give the file the ability to run from the terminal.

So I set the permission as chmod +x to make it executable. From here we get to actually write our code!

First thing I want to do is ping the server, for this I will use the tool subprocess. this will allow us to type a console command are return the results.

Check Ping

import os

import subprocess

hostname = "google.com"

numberof\_pings = 10

pings = subprocess.Popen(["ping","-c " + str(numberof\_pings), hostname], stdout = subprocess.PIPE)

ping\_results = pings.communicate()[0]

cutpoint = ping\_results.find("min/avg/max/stddev =")

minPing = ping\_results[(cutpoint + 21):(cutpoint + 27)]

averagePing = ping\_results[(cutpoint + 28):(cutpoint + 34)]

maxPing = ping\_results[(cutpoint + 35):(cutpoint + 41)]

the subprocess.popen returns our results, but most important to us is the final line round-trip min/avg/max/stddev = 38.642/39.471/40.585/0.471 ms

Once we have that we slice it up to have our min, average and max ping. I realized I need to switch stddev to mdev, and adjust my counts depending on how ping returns. Differences in osx vs ubuntu I guess.

After we have our ping the next item I wanted to capture was my internal and external IP addresses. Internal is what the network has assigned to me, external is what the ISP is assigning to me.

Internal IP:

ifconfig = subprocess.Popen("ifconfig |grep inet", stdout = subprocess.PIPE, shell=True)

ifconfig\_results = ifconfig.communicate()[0]

ifconfig\_cutpoint = ifconfig\_results.find("inet ", 10)

internal\_ip = ifconfig\_results[(ifconfig\_cutpoint+5):(ifconfig\_cutpoint + 17)]

External IP:

external\_ip = subprocess.Popen("dig +short myip.opendns.com @resolver1.opendns.com", stdout = subprocess.PIPE, shell=True)

external\_ip = external\_ip.communicate()[0][0:-1]

For these, we use a very similar process to what we did for our ping. We use some handy command line tools, take the result, pull the data we want out of the string and store it.

Next thing we need is our time formatted in a way the server wants it. I used a one liner for this time = str(datetime.datetime.now().strftime('%Y-%m-%d')) + "T" + str(datetime.datetime.now().time().strftime('%H:%M:%S')) We also need to make sure we include datetime. import datetime

Next, I want to do a speed test. For that I used <https://github.com/sivel/speedtest-cli>.

Speedtest

import speedtest

servers = []

s = speedtest.Speedtest()

s.get\_servers(servers)

s.get\_best\_server()

downspeed = s.download()

upspeed = s.upload()

This is nearly 1:1 copy from their documentation. It gets the best server in the area, and tests up and down speeds from it, then stores them as variables.

The final step to this process is sending it to a server. For this we will use the package, requests The code looks like:

Sending Data to server

import requests

payload = {'node\_name': "http://138.197.216.233:8000/networkconnectivity/networkNodes/536e575d-5e4c-4ddd-b61b-187d072c5aa0/",

'ip\_address': internal\_ip,

'external\_ip': external\_ip,

'timestamp': time,

'ping': averagePing,

'ping\_destination':hostname,

'downspeed': downspeed ,

'upspeed': upspeed

}

r = requests.post('http://138.197.216.233:8000/networkconnectivity/networkData/', auth=('admin', 'testpw'), data=payload)

The first section of this is defining our payload. The node\_name is given to us by the server so it knows which node/pi we are associating these pings to. then we are just linking up all our previous data and building out the data structure for them.

After that, we are posting all this information to our server with some simple authentication.

Now that we have this simple script written we need to move it over to the PI and text it out. I moved the file to /etc/cron.hourly/.

The first thing we do on the pi is install all of our dependencies. (Dig, Requests, and speedtest). Once it tests correctly for running it as a one off its time to hook it up to a cron job. I used the command crontab -e this will open your personal crontab. Inside of this file I used the command \*/15 \* \* \* \* /usr/bin/python /etc/cron.hourly/networknodejob.py to make the tab run every 15 minutes.

And just like that we should be good to go! If you have any questions or advice please reach out! Here is my finished code: <https://github.com/connectivtech/networkbot>