stackoverflow

Ping a site in Python?

The basic code is:

```
from Tkinter import *
import os,sys

ana= Tk()
def ping1():
    os.system('ping')

a=Button(pen)
ip=("192.168.0.1")

a.config(text="PING",bg="white",fg="blue")
a=ping1.ip ???
a.pack()

ana.mainloop()
```

How could I ping a sites or address?

python network-programming beginner

edited Nov 25 '08 at 11:27 xan 2,836 ●7 ●23

asked Nov 25 '08 at 9:58 user40572 71 ●1 ●2

9 Answers

See this pure Python implementation by Matthew Dixon Cowles and Jens Diemer (from the PyLucid CMS sources)

```
import ping, socket
try:
    delay = ping.do_one('www.google.com', timeout=2)
except socket.error, e:
    print "Ping Error:", e
```

answered Nov 25 '08 at 12:39 orip 8,462 ●12 ●28

It's hard to say what your question is, but there are some alternatives.

If you mean to literally execute a request using the ICMP ping protocol, you can get an ICMP library and execute the ping request directly. Google "Python ICMP" to find things like this icmplib. You might want to look at scapy, also.

This will be much faster than using `os.system("ping " + ip )`.

If you mean to generically "ping" a box to see if it's up, you can use the echo protocol on port 7.

For echo, you use the socket library to open the IP address and port 7. You write something on that port, send a carriage return (`"\r\n"`) and then read the reply.

If you mean to "ping" a web site to see if the site is running, you have to use the http protocol on port 80.

For or properly checking a web server, you use urllib2 to open a specific URL. (`/index.html` is always popular) and read the response.

There are still more potential meaning of "ping" including "traceroute" and "finger".

edited **Nov 25 '08 at 11:56** answered **Nov 25 '08 at 11:12**
S.Lott
76.4k ●4 ●67 ●193

You may find Noah Gift's presentation Creating Agile Commandline Tools With Python. In it he combines subprocess, Queue and threading to develop solution that is capable of pinging hosts concurrently and speeding up the process. Below is a basic version before he adds command line parsing and some other features. The code to this version and others can be found here

```
#!/usr/bin/env python2.5
from threading import Thread
import subprocess
from Queue import Queue

num_threads = 4
queue = Queue()
ips = ["10.0.1.1", "10.0.1.3", "10.0.1.11", "10.0.1.51"]
#wraps system ping command
def pinger(i, q):
    """Pings subnet"""
    while True:
        ip = q.get()
        print "Thread %s: Pinging %s" % (i, ip)
        ret = subprocess.call("ping -c 1 %s" % ip,
                        shell=True,
                        stdout=open('/dev/null', 'w'),
                        stderr=subprocess.STDOUT)
        if ret == 0:
            print "%s: is alive" % ip
        else:
            print "%s: did not respond" % ip
        q.task_done()
#Spawn thread pool
for i in range(num_threads):

    worker = Thread(target=pinger, args=(i, queue))
    worker.setDaemon(True)
    worker.start()
#Place work in queue
for ip in ips:
    queue.put(ip)
#Wait until worker threads are done to exit
queue.join()
```

He is also author of: Python for Unix and Linux System Administration

answered **Nov 25 '08 at 12:28**
Ryan Cox
1,094 ●4 ●7

Depending on what you want to achive, you are probably easiest calling the system ping command..

Using the subprocess module is the best way of doing this, although you have to remember the ping command is different on different operating systems!

```
import subprocess

host = "www.google.com"

ping = subprocess.Popen(
    ["ping", "-c", "4", host],
    stdout = subprocess.PIPE,
    stderr = subprocess.PIPE
)

out, error = ping.communicate()
print out
```

You don't need to worry about shell-escape characters. For example..

```
host = "google.com; `echo test`
```

..will **not** execute the echo command.

Now, to actually get the ping results, you could parse the `out` variable. Example output:

```
round-trip min/avg/max/stddev = 248.139/249.474/250.530/0.896 ms
```

Example regex:

```
import re
matcher = re.compile("round-trip min/avg/max/stddev = (\d+.\d+)/(\d+.\d+)/(\d+.\d+
print matcher.match(out).groups()

# ('248.139', '249.474', '250.530', '0.896')
```

Again, remember the output will vary depending on operating system (and even the version of `ping`). This isn't ideal, but it will work fine in many situations (where you know the machines the script will be running on)

edited **Nov 25 '08 at 12:57**
J.F. Sebastian
15.7k ●1 ●23 ●47

answered **Nov 25 '08 at 10:49**
dbr
18k ●1 ●23 ●66

Take a look at Jeremy Hylton's code, if you need to do a more complex, detailed implementation in Python rather than just calling `ping`.

answered **Nov 25 '08 at 11:22**
Brent.Longborough
2,448 ●5 ●19

I use the ping module by Lars Strand. Google for "Lars Strand python ping" and you will find a lot of references.

answered **Nov 25 '08 at 12:41**
akr
984 ●1 ●4 ●11

---

using system ping command to ping a list of hosts:

```
import re
from subprocess import Popen, PIPE
from threading import Thread


class Pinger(object):
    def __init__(self, hosts):
        for host in hosts:
            pa = PingAgent(host)
            pa.start()

class PingAgent(Thread):
    def __init__(self, host):
        Thread.__init__(self)
        self.host = host

    def run(self):
        p = Popen('ping -n 1 ' + self.host, stdout=PIPE)
        m = re.search('Average = (.*)ms', p.stdout.read())
        if m: print 'Round Trip Time: %s ms -' % m.group(1), self.host
        else: print 'Error: Invalid Response -', self.host


if __name__ == '__main__':
    hosts = [
        'www.pylot.org',
        'www.goldb.org',
        'www.google.com',
        'www.yahoo.com',
        'www.techcrunch.com',
        'www.this_one_wont_work.com'
       ]
    Pinger(hosts)
```

answered **Nov 25 '08 at 17:12**
Corey Goldberg
2,536 ●5 ●21

I did something similar this way, as an inspiration:

```
import urllib

def pinger_urllib(host):
  """
  helper function timing the retrival of index.html
  TODO: should there be a 1MB bogus file?
  """
  t1 = time.time()
  urllib.urlopen(host + '/index.html').read()
  return (time.time() - t1) * 1000.0


def task(m):
  """
  the actual task
  """
  delay = float(pinger_urllib(m))
  print '%-30s %5.0f [ms]' % (m, delay)

# parallelization
tasks = []
for m in URLs:
  t = threading.Thread(target=task, args=(m,))
  t.start()
  tasks.append(t)

# synchronization point
for t in tasks:
  t.join()
```

answered **Jul 22 at 12:59**
Harald Schilly
226 ●1 ●7

You can find an updated version of the mentioned script that works on both Windows and Linux at `http://www.g-loaded.eu/2009/10/30/python-ping/`

answered **Nov 16 at 12:10**
Born To Ride
51 ●4

Not the answer you're looking for? Browse other questions tagged python network-programming beginner or ask your own question.