

# *Mawlana Bhashani Science and Technology University*



---

## *Department of Information and Communication Technology*

---

**Course Code** : *ICT-3208*  
**Course Name** : *Network Planning and Designing Lab*  
**Name of the Lab** : *Python for Networking*  
**Lab No.** : *06*

**Submitted to:**  
**Mr. Nazrul Islam**  
**Assistant Professor,**  
**Dept. of ICT, MBSTU**

**Submitted by:**  
**Md. Nazmul Hasan**  
**ID: IT-17005**  
**Session: 2016-17**  
**3rd Year 2nd Semester**

**Date of Submission:** 22-September-2020

## Python for Networking

**Exercise 4.1:** Enumerating interfaces on your machine. Create python scrip using the syntax below (save as list\_network\_interfaces.py):

### Source Code:

```
#!/usr/bin/env python
import sys
import socket
import fcntl
import struct
import array
SIOCGIFCONF = 0x8912 #from C library sockios.h
STUCT_SIZE_32 = 32
STUCT_SIZE_64 = 40
PLATFORM_32_MAX_NUMBER = 2**32
DEFAULT_INTERFACES = 8
def list_interfaces():
    interfaces = []
    max_interfaces = DEFAULT_INTERFACES
    is_64bits = sys.maxsize > PLATFORM_32_MAX_NUMBER
    struct_size = STUCT_SIZE_64 if is_64bits else STUCT_SIZE_32
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    while True:
        bytes = max_interfaces * struct_size
        interface_names = array.array('B', '\0' * bytes)
        sock_info = fcntl.ioctl(sock.fileno(), SIOCGIFCONF, struct.pack('iL',
        bytes, interface_names.buffer_info()[0]))
        outbytes = struct.unpack('iL', sock_info)[0]
        if outbytes == bytes:
            max_interfaces *= 2
        else:
            break
        namestr = interface_names.tostring()
        for i in range(0, outbytes, struct_size):
            interfaces.append((namestr[i:i+16].split('\0', 1)[0]))
    return interfaces
if __name__ == '__main__':
    interfaces = list_interfaces()
    print ("This machine has %s network interfaces: %s."
    %(len(interfaces), interfaces))
```

### Output:

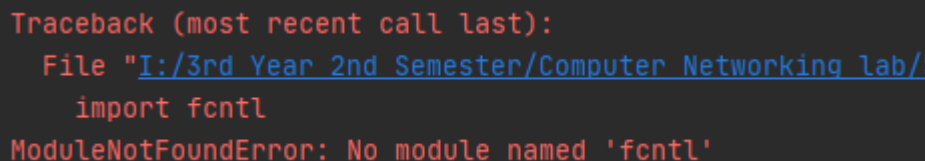
```
Traceback (most recent call last):
  File "I:/3rd Year 2nd Semester/Computer Networking lab/
    import fcntl
ModuleNotFoundError: No module named 'fcntl'
```

**Exercise 4.2:** Finding the IP address for a specific interface on your machine. Create python scrip using the syntax below (save as `get_interface_ip_address.py`):

**Source Code:**

```
import argparse
import sys
import socket
import fcntl
import struct
import array
def get_ip_address(iframe):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915,
    struct.pack('256s', iframe[:15]))[20:24])
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Python networking
    utils')
    parser.add_argument('--iframe', action="store", dest="iframe",
    required=True)
    given_args = parser.parse_args()
    iframe = given_args.iframe
    print ("Interface [%s] --> IP: %s" %(iframe,
    get_ip_address(iframe)))
```

**Output:**



```
Traceback (most recent call last):
  File "I:/3rd Year 2nd Semester/Computer Networking lab/
    import fcntl
ModuleNotFoundError: No module named 'fcntl'
```

**Exercise 4.3:** Finding whether an interface is up on your machine. Create python scrip using the syntax below (save as `find_network_interface_status.py`):

**Source Code:**

```
import argparse
import socket
import struct
import fcntl
import nmap
SAMPLE_PORTS = '21-23'
def get_interface_status(iframe):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    ip_address = socket.inet_ntoa(fcntl.ioctl(sock.fileno(), 0x8915,
    struct.pack('256s', iframe[:15]))[20:24])
```

```

nm = nmap.PortScanner()
nm.scan(ip_address, SAMPLE_PORTS)
return nm[ip_address].state()
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Python networking
utils')
    parser.add_argument('--ifname', action="store", dest="ifname",
required=True)
    given_args = parser.parse_args()
    ifname = given_args.ifname
    print ("Interface [%s] is: %s" %(ifname,
get_interface_status(ifname)))

```

### Output:

```

Traceback (most recent call last):
  File "I:/3rd Year 2nd Semester/Computer Networking lab/
import fcntl
ModuleNotFoundError: No module named 'fcntl'

```

**Exercise 4.4:** Detecting inactive machines on your network. Create python scrip using the syntax below (save as detect\_inactive\_machines.py):

### Source Code:

```

import argparse
import time
import sched
from scapy.layers.inet import sr, srp, IP, UDP, ICMP, TCP, ARP,
Ether
#from scapy.all import sr, srp, IP, UDP, ICMP, TCP, ARP, Ether
RUN_FREQUENCY = 10
scheduler = sched.scheduler(time.time, time.sleep)
def detect_inactive_hosts(scan_hosts):
    """
    Scans the network to find scan_hosts are Live or dead
    scan_hosts can be like 10.0.2.2-4 to cover range.
    See Scapy docs for specifying targets.
    """
    global scheduler
    scheduler.enter(RUN_FREQUENCY, 1, detect_inactive_hosts,
(scan_hosts, ))
    inactive_hosts = []
    try:
        ans, unans = sr(IP(dst=scan_hosts)/ICMP(), retry=0, timeout=1)
        ans.summary(lambda(s,r) : r.sprintf("%IP.src% is alive"))
    for inactive in unans:

```

```

print ("%s is inactive" %inactive.dst)
inactive_hosts.append(inactive.dst)
print ("Total %d hosts are inactive" %(len(inactive_hosts)))
except KeyboardInterrupt:
exit(0)
if __name__ == "__main__":
parser = argparse.ArgumentParser(description='Python networking
utils')
parser.add_argument('--scan-hosts', action="store",
dest="scan_hosts",
required=True)
given_args = parser.parse_args()
scan_hosts = given_args.scan_hosts
scheduler.enter(1, 1, detect_inactive_hosts, (scan_hosts, ))
scheduler.run()

```

### Output:

```

Traceback (most recent call last):
  File "I:/3rd Year 2nd Semester/Computer Networking lab/venv/detect_inactive_machines.py", line 1
    from scapy.layers.inet import sr, srp, IP, UDP, ICMP, TCP, ARP, Ether
ImportError: cannot import name 'srp' from 'scapy.layers.inet' (C:\Users\mnhrh\AppData\Roaming\

```

**Exercise 4.5:** Pinging hosts on the network with ICMP. Create python scrip using the syntax below (save as ping\_remote\_host.py):

### Source Code:

```

#!/usr/bin/env python
import os
import argparse
import socket
import struct
import select
import time
ICMP_ECHO_REQUEST = 8 # Platform specific
DEFAULT_TIMEOUT = 2
DEFAULT_COUNT = 4
class Pinger(object):
    """ Pings to a host -- the Pythonic way """
    def __init__(self, target_host, count=DEFAULT_COUNT,
        timeout=DEFAULT_TIMEOUT):
        self.target_host = target_host
        self.count = count
        self.timeout = timeout
    def do_checksum(self, source_string):
        """ Verify the packet integrity """

```

```

sum = 0
max_count = (len(source_string)/2)*2
count = 0
while count < max_count:
    val = ord(source_string[count + 1])*256 +
    ord(source_string[count])
    sum = sum + val
    sum = sum & 0xffffffff
    count = count + 2
if max_count < len(source_string):
    sum = sum + ord(source_string[len(source_string) - 1])
    sum = sum & 0xffffffff
    sum = (sum >> 16) + (sum & 0xffff)
    sum = sum + (sum >> 16)
    answer = ~sum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer
def receive_pong(self, sock, ID, timeout):
    """
    Receive ping from the socket.
    """
    time_remaining = timeout
    while True:
        start_time = time.time()
        readable = select.select([sock], [], [], time_remaining)
        time_spent = (time.time() - start_time)
        if readable[0] == []: # Timeout
            return
        time_received = time.time()
        recv_packet, addr = sock.recvfrom(1024)
        icmp_header = recv_packet[20:28]
        type, code, checksum, packet_ID, sequence = struct.unpack(
            "bbHHh", icmp_header
        )
        if packet_ID == ID:
            bytes_In_double = struct.calcsize("d")
            time_sent = struct.unpack("d", recv_packet[28:28 +
            bytes_In_double])[0]
            return time_received - time_sent
        time_remaining = time_remaining - time_spent
        if time_remaining <= 0:
            return
    def send_ping(self, sock, ID):
        """
        Send ping to the target host
        """
        target_addr = socket.gethostbyname(self.target_host)
        my_checksum = 0

```

```

# Create a dummy heder with a 0 checksum.
header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, my_checksum,
ID, 1)
bytes_In_double = struct.calcsize("d")
data = (192 - bytes_In_double) * "Q"
data = struct.pack("d", time.time()) + data
# Get the checksum on the data and the dummy header.
my_checksum = self.do_checksum(header + data)
header = struct.pack(
"bbHHh", ICMP_ECHO_REQUEST, 0, socket.htons(my_checksum), ID, 1
)
packet = header + data
sock.sendto(packet, (target_addr, 1))
def ping_once(self):
    """
    Returns the delay (in seconds) or none on timeout.
    """
    icmp = socket.getprotobyname("icmp")
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
    except socket.error, (errno, msg):
        if errno == 1:
            # Not superuser, so operation not permitted
            msg += "ICMP messages can only be sent from root user
processes"
            raise socket.error(msg)
    except Exception, e:
        print "Exception: %s" %(e)
    my_ID = os.getpid() & 0xFFFF
    self.send_ping(sock, my_ID)
    delay = self.receive_pong(sock, my_ID, self.timeout)
    sock.close()
    return delay
def ping(self):
    """
    Run the ping process
    """
    for i in xrange(self.count):
        print "Ping to %s..." % self.target_host,
        try:
            delay = self.ping_once()
        except socket.gaierror, e:
            print "Ping failed. (socket error: '%s')" % e[1]
            break
        if delay == None:
            print "Ping failed. (timeout within %ssec.)" % self.timeout
        else:
            delay = delay * 1000
            print "Get pong in %0.4fms" % delay

```

```

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Python ping')
    parser.add_argument('--target-host', action="store",
dest="target_host", required=True)
    given_args = parser.parse_args()
    target_host = given_args.target_host
    pinger = Pinger(target_host=target_host)
    pinger.ping()

```

### Output:

```

File "I:/3rd Year 2nd Semester/Computer Networking lab/venv/ping_remote_host.py", line 81
    except socket.error as (errno, msg):
                        ^
SyntaxError: invalid syntax

```

**Exercise 4.6:** Pinging hosts on the network with ICMP using pc resources. Create python scrip using the syntax below (save as ping\_subprocess.py):

### Source Code:

```

import subprocess
import shlex
command_line = "ping -c 1 10.0.1.135"
if __name__ == '__main__':
    args = shlex.split(command_line)
    try:
        subprocess.check_call(args, stdout=subprocess.PIPE, stderr=subprocess.
PIPE)
    print ("Your pc is up!")
    except subprocess.CalledProcessError:
    print ("Failed to get ping.")

```

### Output:

```

Failed to get ping.

Process finished with exit code 0

```

**Exercise 4.7:** Scanning the broadcast of packets. Create python scrip using the syntax below (save as broadcast\_scanning.py):

### Source Code:

```

from scapy import all
from scapy.layers.inet import sr, srp, IP, UDP, ICMP, TCP, ARP,

```



```

Ether,
sniff
captured_data = dict()
END_PORT = 1000
def monitor_packet(pkt):
    if IP in pkt:
        if not captured_data.has_key(pkt[IP].src):
            captured_data[pkt[IP].src] = []
        if TCP in pkt:
            if pkt[TCP].sport <= END_PORT:
                if not str(pkt[TCP].sport) in captured_data[pkt[IP].src]:
                    captured_data[pkt[IP].src].append(str(pkt[TCP].sport))
            os.system('clear')
            ip_list = sorted(captured_data.keys())
            for key in ip_list:
                ports=', '.join(captured_data[key])
                if len(captured_data[key]) == 0:
                    print ('%s' % key)
                else:
                    print ('%s (%s)' % (key, ports))
if __name__ == '__main__':
    sniff(prn=monitor_packet, store=0)

```

### Output:

```

Traceback (most recent call last):
  File "I:/3rd Year 2nd Semester/Computer Networking lab/venv/broadcast_scanning.py", line 2,
    from scapy.layers.inet \
ImportError: cannot import name 'srp' from 'scapy.layers.inet' (C:\Users\mnhr\AppData\Roaming
Process finished with exit code 1

```

### **Exercise 4.8:** Sniffing packets on your network

**Tcpdump** is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. Distributed under the BSD license,[3] tcpdump is free software.

- Open a linux terminal and check the usage of tcpdump using the command line  
tcpdump -help
- Using tcpdump get the traffic present in the Ethernet interface of your pc (10 packet only), which is the command line?
- Using the subprocess write a program for sniffing 1 packet of the Ethernet interface? (Save as packet\_sniffer.py).

No Solution

**Exercise 4.9:** Performing a basic Telnet. Create python scrip using the syntax below (save as echo\_client.py):

### Source Code:

```
#!/usr/bin/env python
import socket
TCP_IP = '127.0.0.1'
TCP_PORT = 62
BUFFER_SIZE = 20 # Normally 1024, but we want fast response
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)
conn, addr = s.accept()
print ('Connection address:', addr)
while 1:
    data = conn.recv(BUFFER_SIZE)
    if not data: break
    print ("received data:", data)
    conn.send(data) # echo
    conn.close()
```

### **Output:**

No output

**Conclusion:** In this Lab 1<sup>st</sup> three problems cannot be solved because of no importing 'fcntl'. In other problems, I have tried my best to solve this problem but I could not succeed. So, most of the problem I cannot solve from this lab.