Try experiment in mininet

You can try to solve this problem by simulating in mininet first, we'll post the way how to use the real server when the server is ready asap. Finally, you'll still need to accomplish this task in real environment.

Resource:

1. **17.2. socket — Low-level networking interface** **https://docs.python.org/2/library/socket.html**
2. **Socket Programming HOWTO** **https://docs.python.org/2/howto/sockets.html**

Here are four minimal example programs using the TCP/IP protocol: a server that echoes all data that it receives back (servicing only one client), and a client using it.
Note that a server must perform the sequence socket(), bind(), listen(), accept() (possibly repeating the accept() to service more than one client), while a client only needs the sequence socket(), connect(). Also note that the server does not sendall()/recv() on the socket it is listening on but on the new socket returned by accept().

The first two examples support IPv4 only.

```
# Echo server program
import socket

HOST = ''                 # Symbolic name meaning all available
interfaces
PORT = 50007              # Arbitrary non-privileged port
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Connected by', addr
while 1:
    data = conn.recv(1024)
    if not data: break
    conn.sendall(data)
conn.close()
```

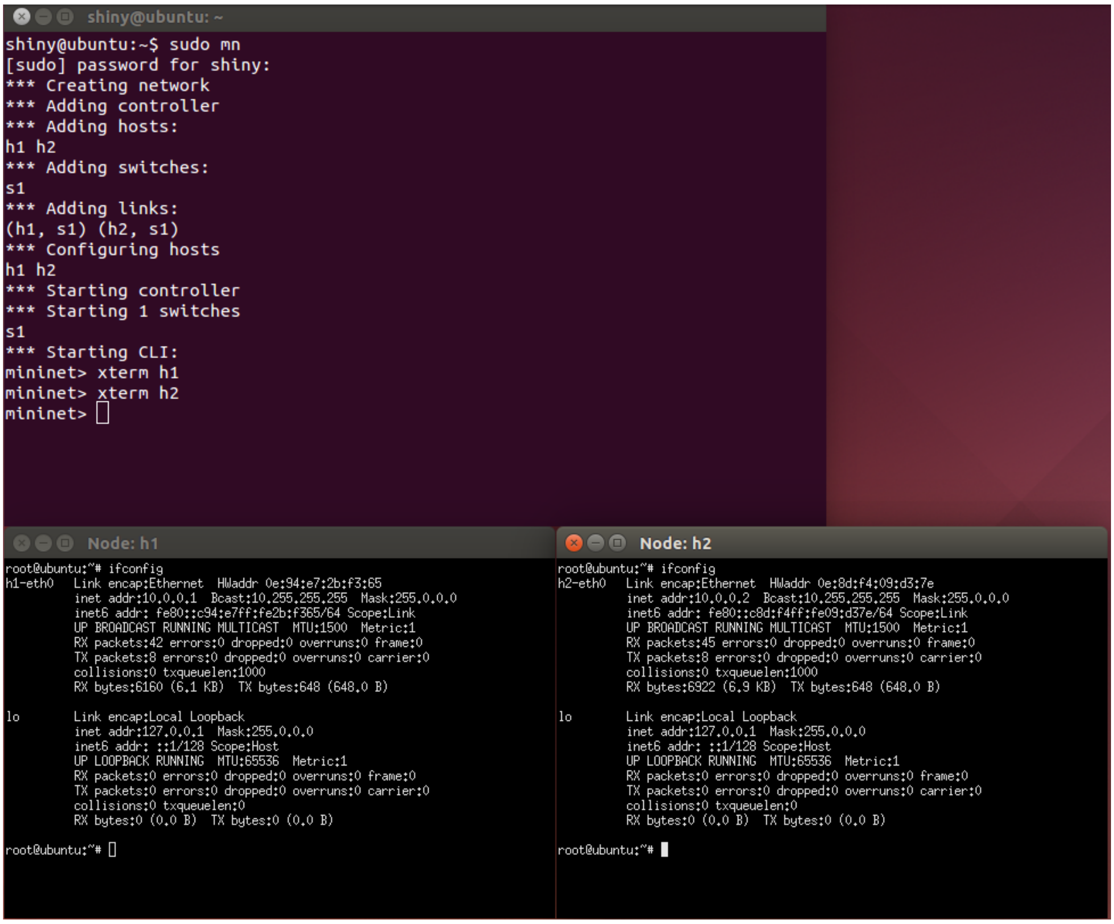
```
# Echo client program
import socket

HOST = '10.0.0.1'          # According to your server ip address
PORT = 50007               # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.sendall('Hello, world')
data = s.recv(1024)
s.close()
print 'Received', repr(data)
```

Step 1:

mininet>>xterm h1
mininet>>xterm h2

Step2:

Command “ifconfig” to check the info of network interface



```
shiny@ubuntu:~$ sudo mn
[sudo] password for shiny:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet> xterm h1
mininet> xterm h2
mininet> 
```

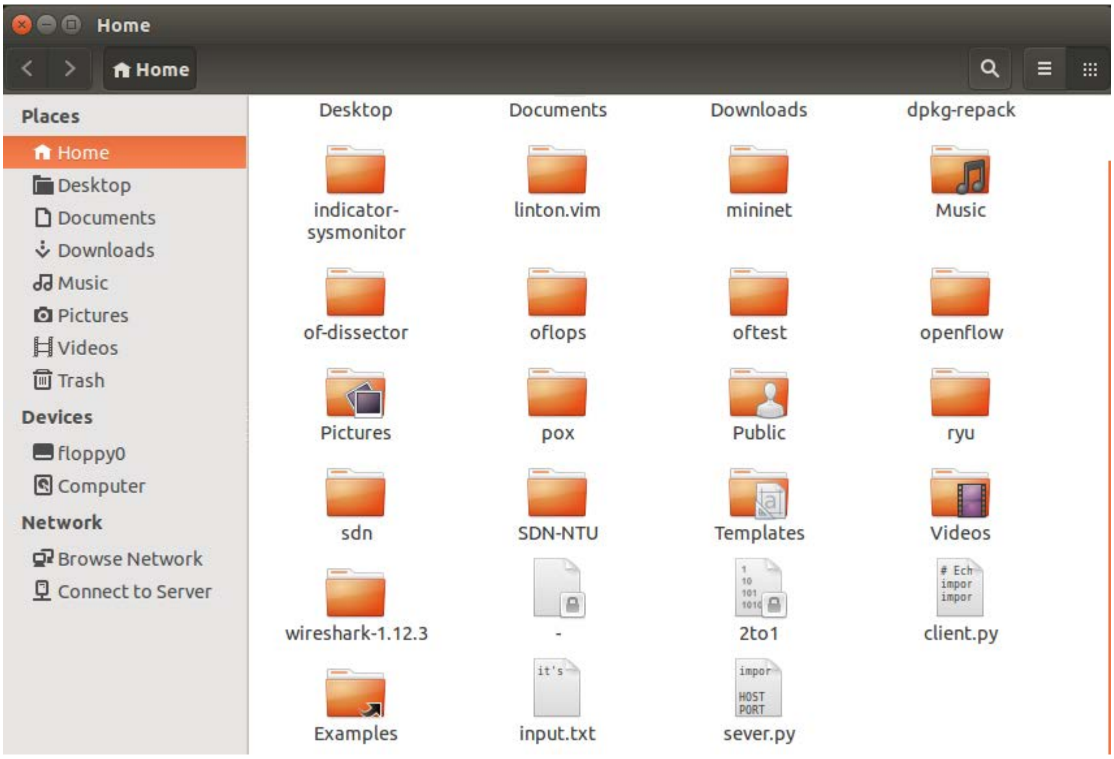
```
root@ubuntu:~# ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 0e:94:e7:2b:f3:65
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::c94:e7ff:fe2b:f365/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6160 (6.1 KB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:~# 
```

```
root@ubuntu:~# ifconfig
h2-eth0   Link encap:Ethernet  HWaddr 0e:8d:f4:09:d3:7e
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::c8d:f4ff:fe09:d37e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6922 (6.9 KB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:~# 
```

h1_ip = 10.0.0.1
h2_ip = 10.0.0.2

Step3 :

Place your python code in home directory or “cd” to where you place your code



Step 4 :

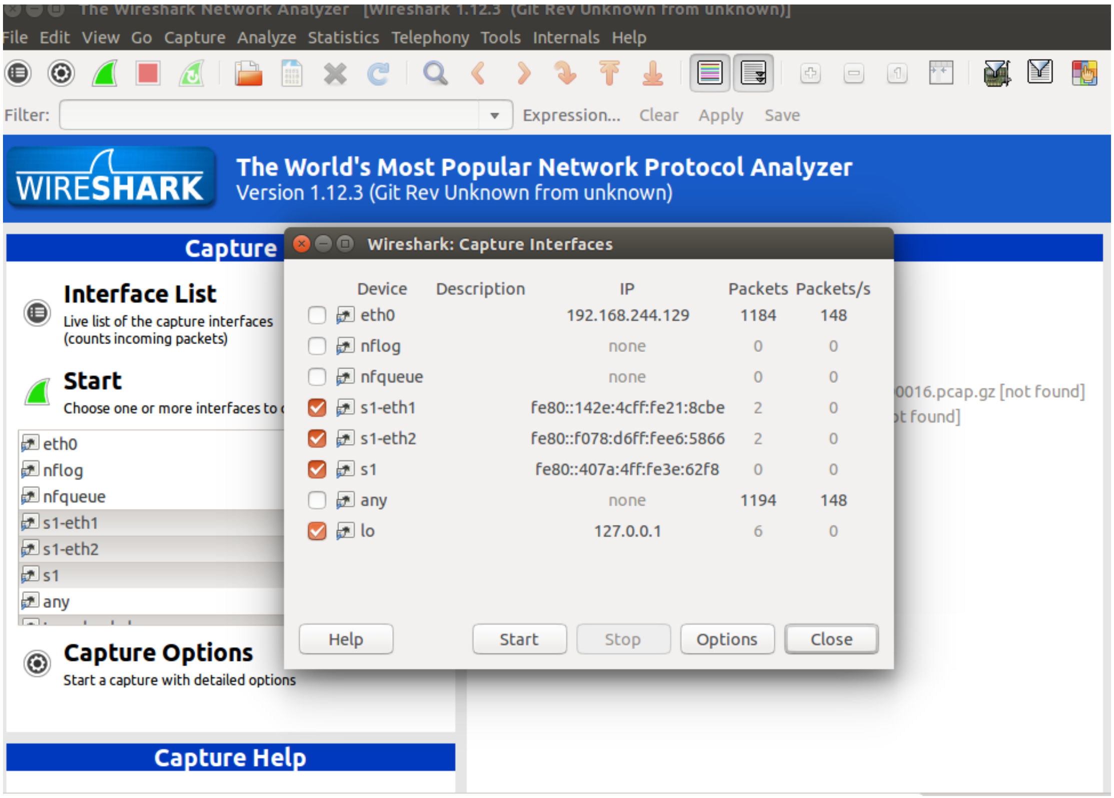
Google “how to read and store the txt file python”

Basic way:

```
file = open('input.txt', 'r')
content = file.read()
print(content)
```

Step 5 :

sudo wireshark
To observe the packets when socket transmitting


The Wireshark Network Analyzer [Wireshark 1.12.3 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
WIRESHARK
Capture
Capture Help
Wireshark: Capture Interfaces
Help Start Stop Options Close

The World's Most Popular Network Protocol Analyzer
Version 1.12.3 (Git Rev Unknown from unknown)

Interface List
Live list of the capture interfaces
(counts incoming packets)

Start
Choose one or more interfaces to c

eth0
nflog
nfqueue
s1-eth1
s1-eth2
s1
any

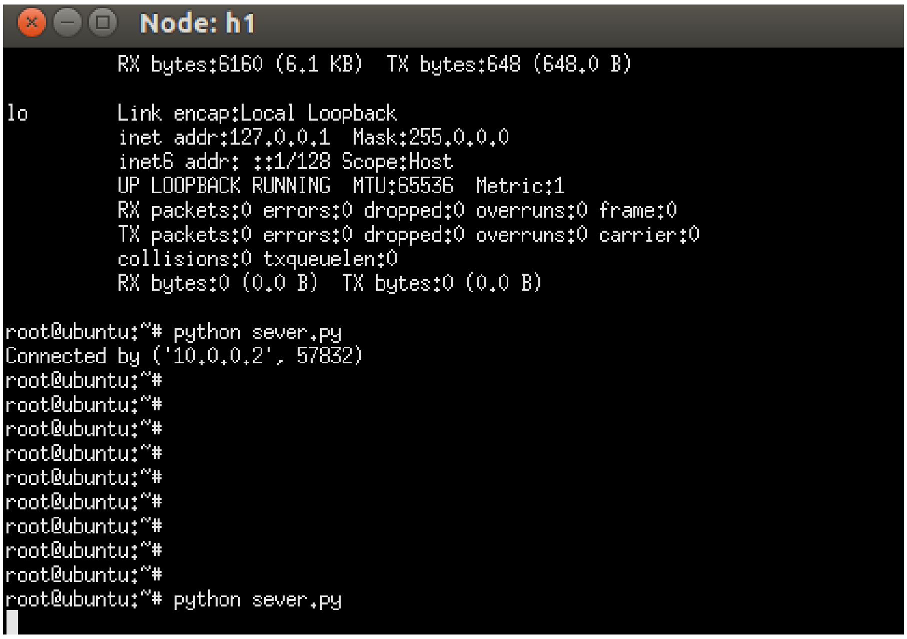
Capture Options
Start a capture with detailed options

0016.pcap.gz [not found]
ot found]

	Device	Description	IP	Packets	Packets/s
□	eth0		192.168.244.129	1184	148
□	nflog		none	0	0
□	nfqueue		none	0	0
☑	s1-eth1		fe80::142e:4cff:fe21:8cbe	2	0
☑	s1-eth2		fe80::f078:d6ff:fee6:5866	2	0
☑	s1		fe80::407a:4ff:fe3e:62f8	0	0
□	any		none	1194	148
☑	lo		127.0.0.1	6	0

Step 6 :

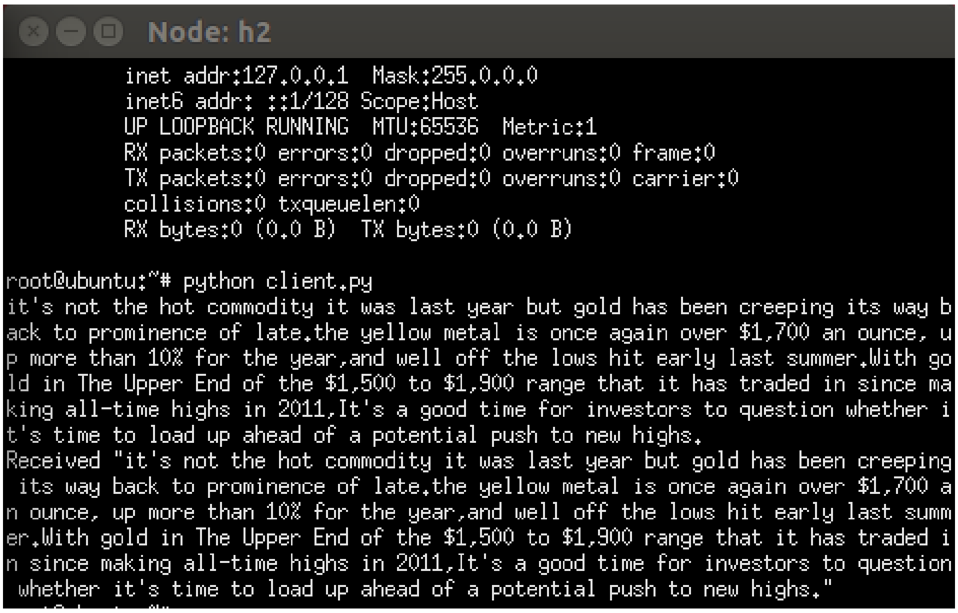
python server.py
(run server code in your sever first to listen to the requests)



```
          RX bytes:6160 (6.1 KB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:~# python sever.py
Connected by ('10.0.0.2', 57832)
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# python sever.py
```

Step 7:

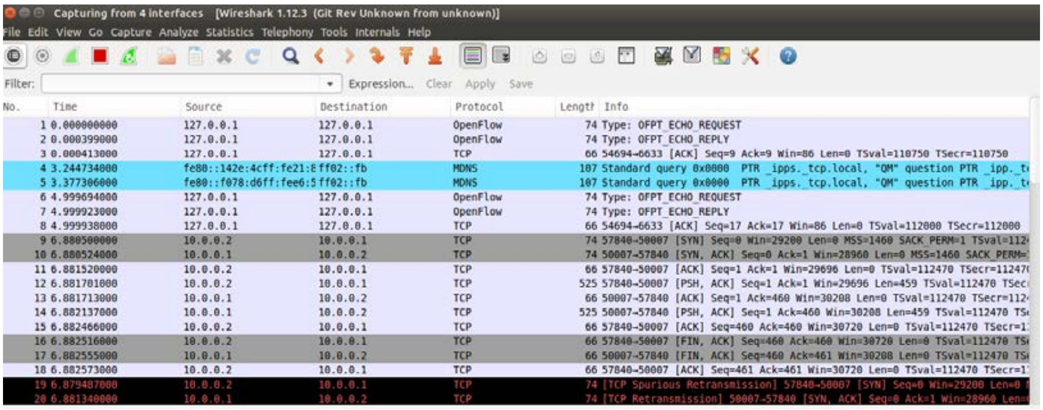
python client.py



```
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:~# python client.py
it's not the hot commodity it was last year but gold has been creeping its way b
ack to prominence of late.the yellow metal is once again over $1,700 an ounce, u
p more than 10% for the year,and well off the lows hit early last summer.With go
ld in The Upper End of the $1,500 to $1,900 range that it has traded in since ma
king all-time highs in 2011,It's a good time for investors to question whether i
t's time to load up ahead of a potential push to new highs.
Received "it's not the hot commodity it was last year but gold has been creeping
 its way back to prominence of late.the yellow metal is once again over $1,700 a
n ounce, up more than 10% for the year,and well off the lows hit early last summ
er.With gold in The Upper End of the $1,500 to $1,900 range that it has traded i
n since making all-time highs in 2011,It's a good time for investors to question
 whether it's time to load up ahead of a potential push to new highs."
```

Step 8 :

Observe the process of socket communication and write what you notice in your report.


Capturing from 4 interfaces [Wireshark 1.12.3 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
2	0.000399000	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
3	0.000413000	127.0.0.1	127.0.0.1	TCP	66	54694→6633 [ACK] Seq=9 Ack=9 Win=86 Len=0 TSval=110750 TSecr=110750
4	3.244734000	fe80::142e:4cff:fe21:8	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._t
5	3.377306000	fe80::f078:d6ff:fee6:5	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._t
6	4.999694000	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
7	4.999923000	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
8	4.999938000	127.0.0.1	127.0.0.1	TCP	66	54694→6633 [ACK] Seq=17 Ack=17 Win=86 Len=0 TSval=112000 TSecr=112000
9	6.880500000	10.0.0.2	10.0.0.1	TCP	74	57840→50007 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=112
10	6.880524000	10.0.0.1	10.0.0.2	TCP	74	50007→57840 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=
11	6.881520000	10.0.0.2	10.0.0.1	TCP	66	57840→50007 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=112470 TSecr=11247
12	6.881701000	10.0.0.2	10.0.0.1	TCP	525	57840→50007 [PSH, ACK] Seq=1 Ack=1 Win=29696 Len=459 TSval=112470 TSec
13	6.881713000	10.0.0.1	10.0.0.2	TCP	66	50007→57840 [ACK] Seq=1 Ack=460 Win=30208 Len=0 TSval=112470 TSecr=112
14	6.882137000	10.0.0.1	10.0.0.2	TCP	525	50007→57840 [PSH, ACK] Seq=1 Ack=460 Win=30208 Len=459 TSval=112470 TS
15	6.882466000	10.0.0.2	10.0.0.1	TCP	66	57840→50007 [ACK] Seq=460 Ack=460 Win=30720 Len=0 TSval=112470 TSecr=1
16	6.882516000	10.0.0.2	10.0.0.1	TCP	66	57840→50007 [FIN, ACK] Seq=460 Ack=460 Win=30720 Len=0 TSval=112470 TS
17	6.882555000	10.0.0.1	10.0.0.2	TCP	66	50007→57840 [FIN, ACK] Seq=460 Ack=461 Win=30208 Len=0 TSval=112470 TS
18	6.882573000	10.0.0.2	10.0.0.1	TCP	66	57840→50007 [ACK] Seq=461 Ack=461 Win=30720 Len=0 TSval=112470 TSecr=1
19	6.879487000	10.0.0.2	10.0.0.1	TCP	74	[TCP Spurious Retransmission] 57840→50007 [SYN] Seq=0 Win=29200 Len=0
20	6.881340000	10.0.0.1	10.0.0.2	TCP	74	[TCP Retransmission] 50007→57840 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=

Step 9 :

Write the “correcting txt-file function code” to make the format of input.txt fit the constraints.

Assignment

Warning: Don’t plagiarize what already on the Internet, it’ll make you lose 40% points at least.

Hand in the following files in one rar.

1. Your codes, including server.py and client.py

Which could make format of any input txt-file correct.
It's not the hot commodity it was last year but gold has been creeping its way back to prominence of late.
The yellow metal is once again over $1,700 an ounce, up more than 10% for the year, and well off the lows hit early last summer.
With gold in the upper end of the $1,500 to $1,900 range that it has traded in since making all-time highs in 2011, it's a good time for investors to question whether it's time to load up ahead of a potential push to new highs.

Total count : 92 words, 463 characters

2. Report, what you observe through wireshark during the experiment such as, the tcp window size.