Testing Environment

Server IP: 192.168.0.15

Part I

Source code:

```
from socket import *
import time
serverName = '192.168.0.15'
serverPort = 12000

clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
message='lower case'
result=[[0 for i in xrange(25)] for i in xrange(5)]
for m in xrange(5):
    print 'm=%s for this round' % (m+1,)
    clientSocket.settimeout(m+1)
    for n in xrange(25):
        valid_count=0
        for i in xrange(n+1):
            try:
                start_time=time.time()
                print 'clientPing_v1 %s %s %s'  % (n+1,i+1,start_time)
                clientSocket.sendto(message.encode(), (serverName,
serverPort))
                modifiedMessage, serverAddress =
clientSocket.recvfrom(2048)
                end_time=time.time()
                delta=end_time-start_time
                result[m][n]+=delta
                valid_count+=1
                print "message from server: " + modifiedMessage.decode()
                print "RTT: %f" % (delta,)
            except timeout as e:
                print 'Request time out'
```

```
        if valid_count:
            result[m][n]/=valid_count
for r in result:
    print r
clientSocket.close()
```

A screenshot of runs

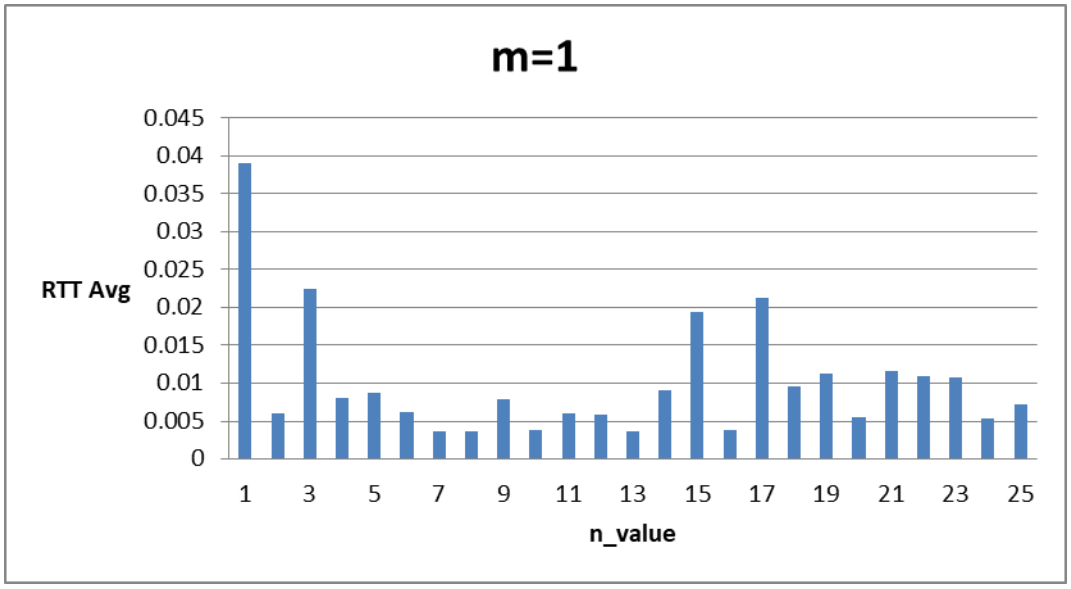
Starting the client

```
\assignments\assignment2>python client.py >output.txt
```

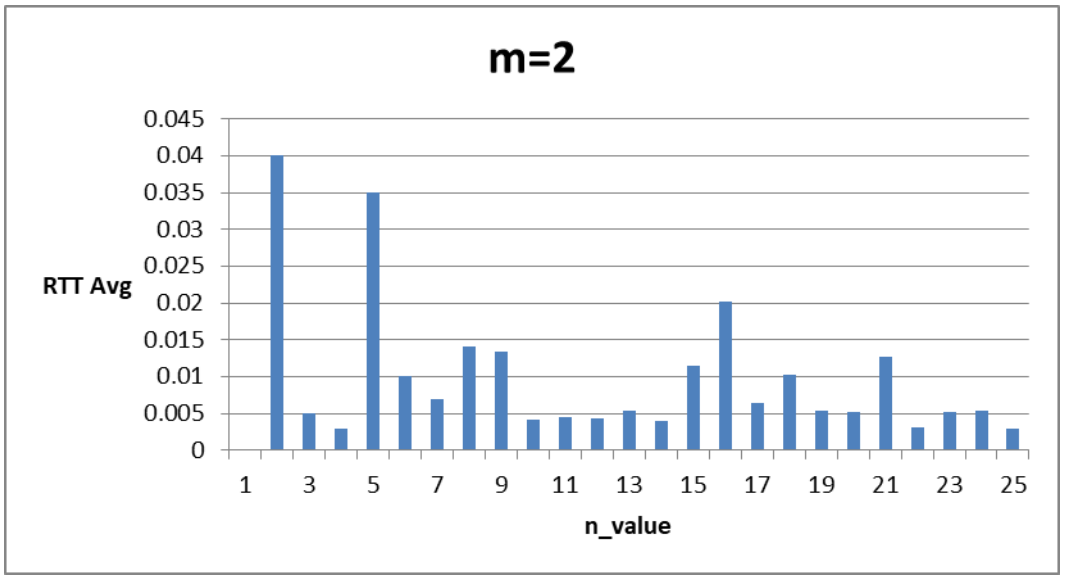
Since the output is too long, it was directed to a text file

```
m=1 for this round
clientPing_v1 1 1 1518912822.41
message from server: LOWER CASE
RTT: 0.039000
clientPing_v1 2 1 1518912822.45
Request time out
clientPing_v1 2 2 1518912823.45
message from server: LOWER CASE
RTT: 0.006000
clientPing_v1 3 1 1518912823.46
message from server: LOWER CASE
RTT: 0.017000
clientPing_v1 3 2 1518912823.47
Request time out
clientPing_v1 3 3 1518912824.47
message from server: LOWER CASE
RTT: 0.028000
clientPing_v1 4 1 1518912824.5
message from server: LOWER CASE
RTT: 0.019000
clientPing_v1 4 2 1518912824.52
message from server: LOWER CASE
RTT: 0.003000
clientPing_v1 4 3 1518912824.52
message from server: LOWER CASE
```

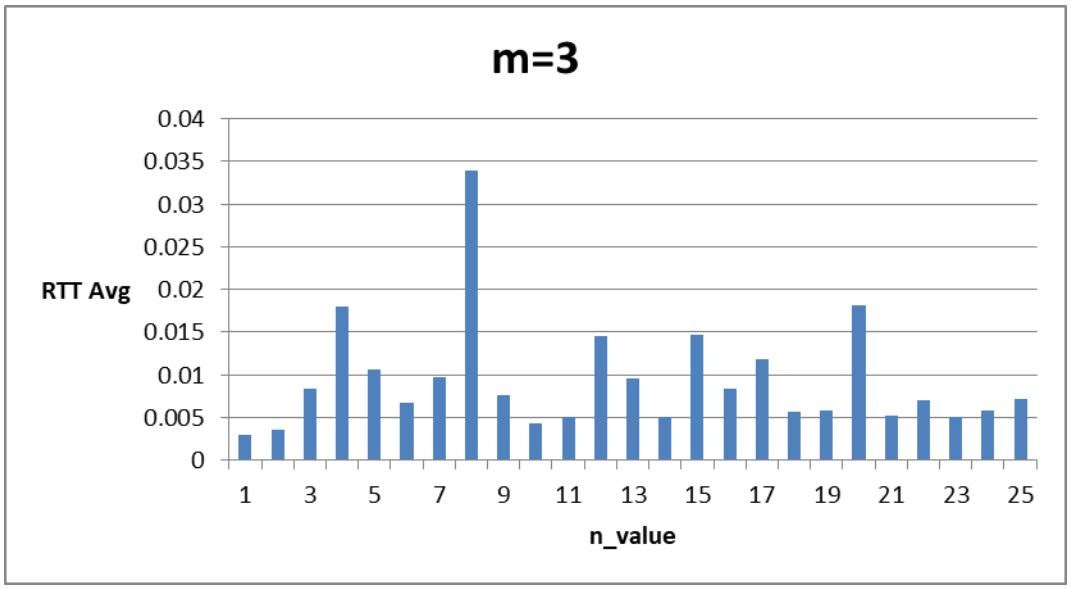
Five graphs



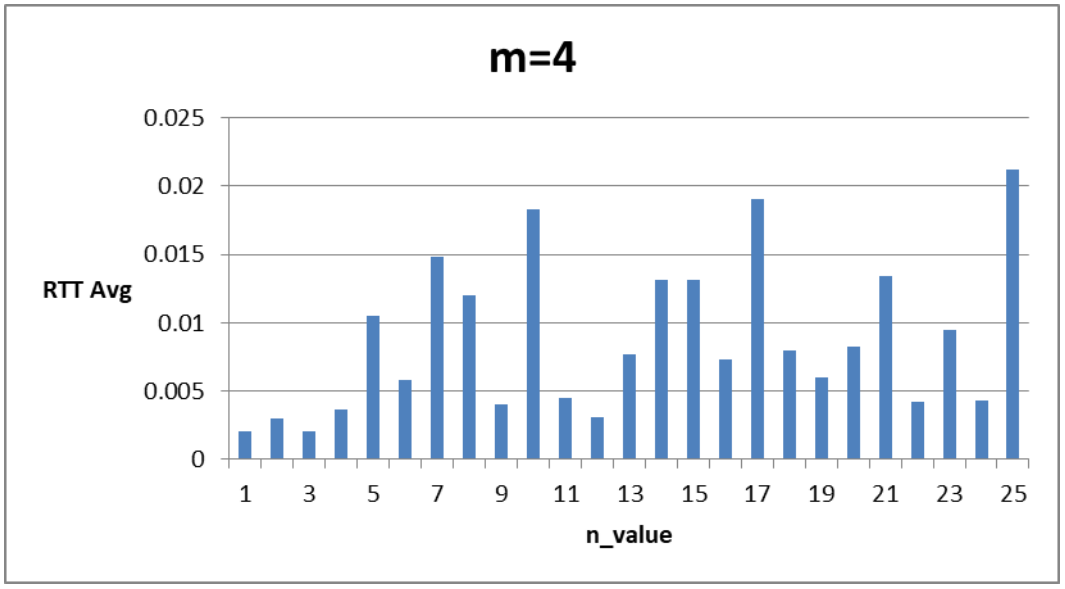
Number of lost messages: 84



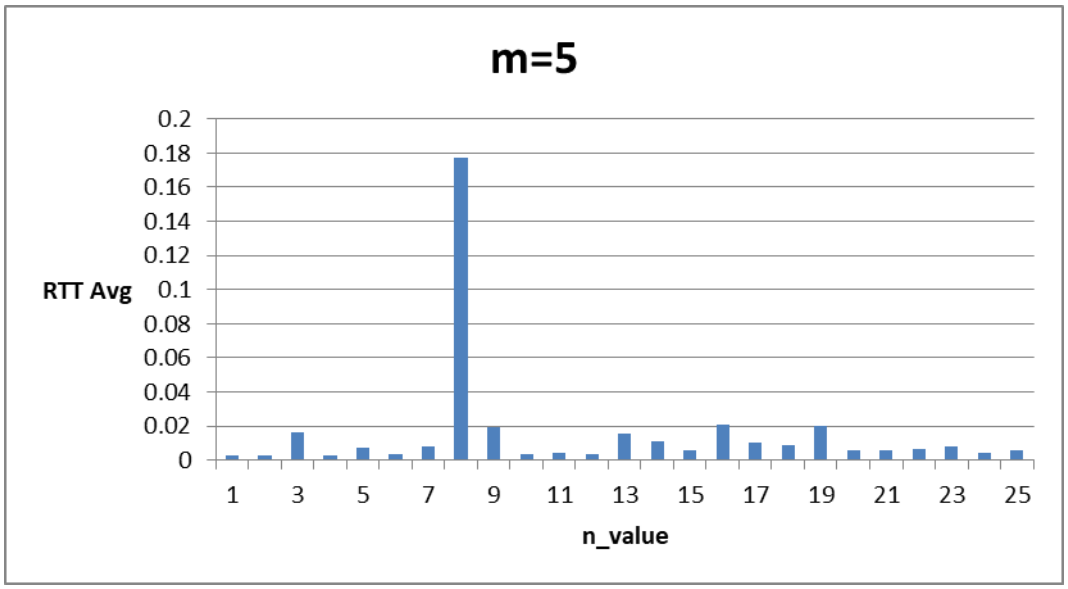
Number of lost messages: 86



Number of lost messages: 91



Number of lost messages: 104



Number of lost messages: 91

Discussion of the result

According to the 5 graphs, we can see that:

1. The number of messages lost is not related to the timeout value
2. RRTs are quite small within a subnet, mostly less than 200ms
3. Connection check and retry mechanism are crucial for any internet softwares, whether server-side or client-side, especially when servers and clients are located in different spots geographically.