CS356 Programming Assignment

Phase 1

1. Short description of your protocol including message format, message types and message flow
   1. The protocol used here is UDP and using SOCK\_DGRAM family the message sending is a dictionary, after that I flattened to encode as string.
2. Code listing of Client and Server
   1. # UDPServer.py  
        
      **import** socket  
      **import** sys  
      **import** random  
      **import** datetime  
        
        
      **def Main**():  
       HOST = '128.235.208.225'  
       PORT = 12346  
       MAX = 4096  
        
       # Datagram (udp) socket  
       **try**:  
       s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
       print('Socket created')  
       **except** socket.error **as** msg:  
       print('Failed to create socket. Error Code : ' + str(msg[0]) + ' Message ' + msg[1])  
       sys.exit()  
        
       # Bind the socket to the port  
       server\_address = (HOST, PORT)  
        
       **try**:  
       s.bind(server\_address)  
       **except** socket.error **as** msg:  
       print(msg)  
       print('Socket Bind completed')  
       # print('Listening at', s.getsockname())  
       print('starting up on {} port {}'.format(\*server\_address))  
       message =['Requested One here my Information:', 'We received your headers']  
        
       first\_msg = message[0]  
       second\_message = message[1]  
        
       **while True**:  
       # Generate random number in the range of 0 to 10  
       rand = random.randint(0, 10)  
       current\_time = datetime.datetime.now().time()  
       print('\nwaiting to receive message...')  
       data, address = s.recvfrom(MAX)  
        
       **if** data == b'1':  
       print("requested number:".upper(), data.decode('UTF-8'), 'from ROUTER0 address: ', address)  
       print(current\_time, 'received {} bytes from {}'.format(len(data), address))  
       print('Sending message to: ', address)  
       sent = s.sendto(first\_msg.encode(), address)  
       print(current\_time, 'sent {} bytes back to {}'.format(sent, address))  
       **elif** data == b'2':  
       print("requested number:".upper(), data.decode('UTF-8'), 'from address: ', address)  
       sent = s.sendto(second\_message.encode(), address)  
       print(current\_time, 'sent {} bytes back to {}'.format(sent, address))  
       new\_data, same\_address = s.recvfrom(MAX)  
       print('ROUTER0 Header information'.upper(), new\_data.decode(), 'FROM:', same\_address)  
       header\_info = s.sendto(new\_data, same\_address)  
       print('received {} bytes from {}'.format(len(new\_data), same\_address))  
        
       print(current\_time, 'received {} bytes from {}'.format(len(data), address))  
       print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
        
      **if** \_\_name\_\_ == '\_\_main\_\_':  
       Main()
   2. Client

# UDPClient.py  
**import** socket  
**import** time  
**from** datetime **import** timedelta  
  
HOST = '128.235.208.225'  
PORT = 12346  
MAX = 4096  
  
  
  
# Creating UDP socket  
s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
  
#Bind the socket to the port  
server\_address = (HOST, PORT)  
message = bytes(str(1), "ascii")  
header = bytes(str(2), "ascii")  
my\_header\_info = {'Source\_Router': 0, 'Router0': 1, 'Router1': 3, 'Router2': 7}  
separator = ","  
completed\_header = separator.join("=".join((str(k), str(v))) **for** k, v **in** my\_header\_info.items())  
  
start = time.time()  
  
**try**:  
 print("================================================================")  
 print('Sending request number: ', message.decode())  
 print("================================================================")  
 print('SOURCE\_ROUTER', server\_address)  
 sent = s.sendto(message, server\_address)  
 end = time.time()  
 elapsed\_time = end - start  
  
 #receive response  
 data, address = s.recvfrom(MAX)  
 print('from client '.upper(), s.getsockname())  
 print('from server: {!r}'.format(data.decode()).upper(), address)  
 print('Time Elapse ', str(timedelta(seconds=elapsed\_time)))  
 print("================================================================")  
**finally**:  
 print('Sending heading information')  
 my\_headers = s.sendto(header, server\_address)  
 s.sendto(completed\_header.encode(), server\_address)  
 print('Source Router sending header: ', header.decode())  
 modifiedMessage, serverAddress = s.recvfrom(MAX)  
 print('From Router1 say : {!r}'.format(modifiedMessage.decode()).upper(), serverAddress)  
 modifiedMessage\_2, serverAddress\_2 = s.recvfrom(MAX)  
 print('From Router1 say you sent : {!r}'.format(modifiedMessage\_2.decode()).upper(), serverAddress\_2)  
 print('closing socket!'.upper())  
 s.close()

1. Screenshot showing that the code compiled on the target machine successfully without errors

|  |
| --- |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.27.29 PM.png |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.27.41 PM.png |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.28.03 PM.png |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.28.33 PM.png |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.28.45 PM.png |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.40.26 PM.png |
| /Users/davidguardia/Desktop/Screen Shot 2017-03-22 at 11.29.23 PM.png |

1. Screenshot of output described above (client and server routing tables)
2. Evidence that messages are being exchanged across the network. This may include using a tool such as wireshark and/or screenshots of trace messages in your program to indicate messages sent and received, including information such as message type, data being sent/received, hostname, IP address and ports used, and timestamps.

List References consulted when designing and building your program (if any). Clearly indicate what code is yours versus based on others.