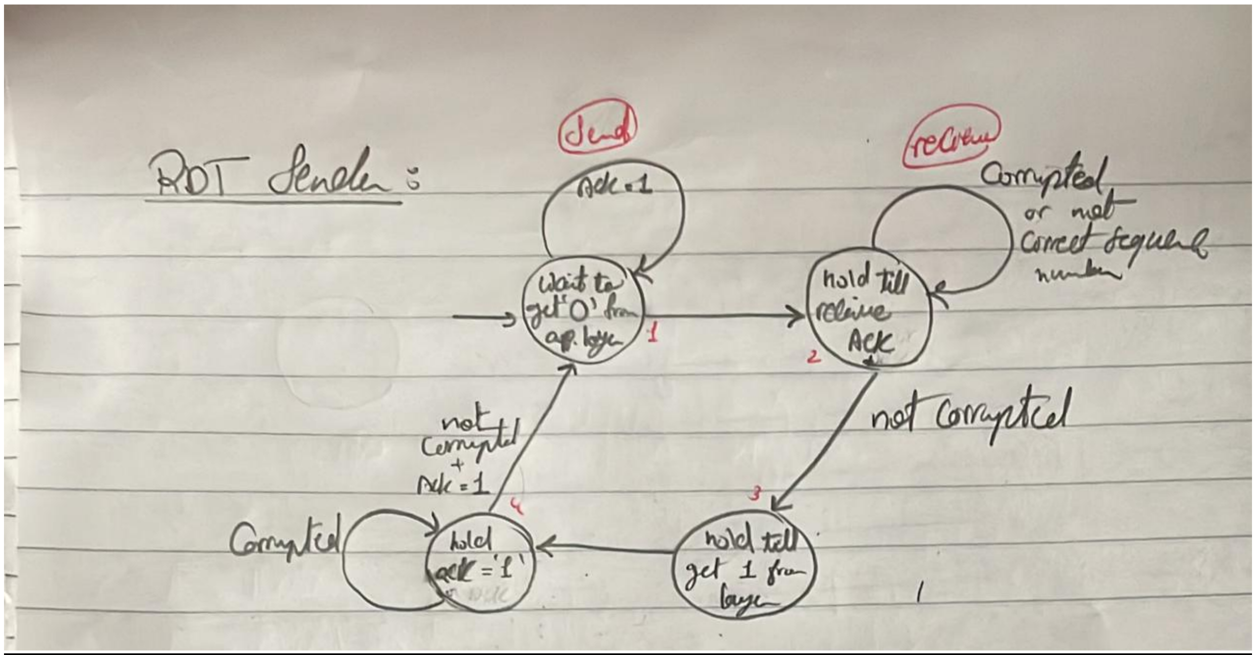
Project 1 Report

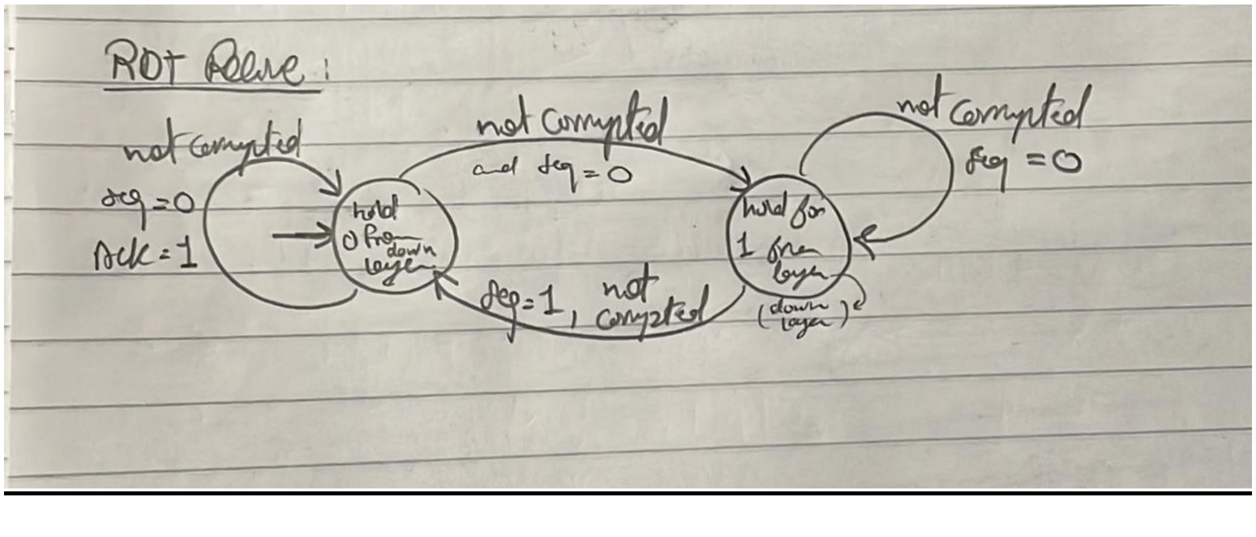
Team member 1 : Habiba Mohamed Fathalla (T-05) (55-25006) (habiba.gaber)

Team member 2 : Arwa Hussein (T-05) (55-0406) (arwa.hussein)

The code was done together with equal contribution as well as the report contents

Finite State Machine





<u>Changes in skeleton code:</u>

<u>In Sender.py :</u>

```
def get_checksum(data):
       if data is not None :
          return ord(data)
       else :
          return
```

<u>Explanation :</u>

To get the Ascii code of provided data using the ord() function

——

```
def clone_packet(packet):
        pkt_clone = {
            'sequence_number': packet['sequence_number'],
          'data': packet['data'],
          'checksum': packet['checksum']
         }
         return pkt_clone
```

<u>Explanation :</u>

Used to make a clone of a given packet

——

```
def is_corrupted(reply):
if reply is None :
         return False
elif reply['checksum'] != ord(reply[‘ack']) :
         return True
else :
         return False
```

<u>Explanation :</u>

Checks if a reply is corrupted or not. If the provided reply is Null, it also return False.

——

```
def is_expected_seq(reply, exp_seq):
         if reply['ack'] ==  exp_seq :
            return True
          else :
             return False
```

<u>Explanation :</u>

Checks if the received reply has the same sequence number as the expected sequence

——————————————————————————————————————

```
def rdt_send(self, process_buffer):
    for data in process_buffer:

        checksum = RDTSender.get_checksum(data)
        pkt = RDTSender.make_pkt(self.sequence, data, checksum)

        print ('Sender expecting seq_num: ' +self.sequence)
        print('Sender sending : ', pkt)

        clone = RDTSender.clone_packet(pkt)
        reply = self.net_srv.udt_send(clone)

        while RDTSender.is_corrupted(reply) == True or
                RDTSender.is_expected_seq(reply,self.sequence) == False :

            checksum = RDTSender.get_checksum(data)
            pkt = RDTSender.make_pkt(self.sequence, data, checksum)
            clone= RDTSender.clone_packet(pkt)
            reply = self.net_srv.udt_send(clone)

        if self.sequence == '0' :
            self.sequence = '1'
        else :
            self.sequence = ‘0'
```

Explanation :

A for loop was used to loop over each data in the process buffer. We initialized a 'checksum' variable; where we placed the Ascii code of the singular data, a 'pkt' variable, to create a new packet with the make_pkt() function, using the sequence number (initially = '0') , the data and its checksum. We also initialized a new variable called 'clone'; where we placed a clone of the created packet, using the clone_packet() function. Then we initialized a 'reply' variable where we use to send a clone of the packet to the receiver. We created a while loop to check if reply is corrupted or if the sequence is not correct, if so, we again initialized the above variables, until a non-corrupted, correct sequence number packet and clone is created, then we start sending the correct reply using the correct clone. After that, we alternate the sequence number, if the sequence number of the previous packet is '0', the sequence number of the next packet will be '1', and vice versa.

In Receiver.py :

```
def is_corrupted(packet):
        result = ord(packet['data'])
        if packet['checksum'] != result :
                return True
        else :
                return False
```

Explanation :

Get ascii code of data in the provided packet then check if the checksum of the provided packet is equal to the Ascii code of data in the same packet, if function returns true it means the packet is corrupted otherwise it is not corrupted

——

```
def is_expected_seq(rcv_pkt, exp_seq):
        if  rcv_pkt['sequence_number'] == exp_seq:
              return True
        else :
              return False
```

Explanation :

Check if the sequence number of the received packet is equal to the expected sequence number

———————————————————————————————————————

```
def rdt_rcv(self, rcv_pkt):
      print('Receiver expecting seq_num : ' + self.sequence)
if not RDTReceiver.is_corrupted(rcv_pkt) and RDTReceiver.is_expected_seq(rcv_pkt,
self.sequence):
         data_recieved = rcv_pkt['data']
         ReceiverProcess.deliver_data(data_recieved)
         checksum=ord(self.sequence)
         reply_pkt = RDTReceiver.make_reply_pkt(self.sequence,checksum)
         self.sequence = '0' if self.sequence == '1' else '1'
      else :
         if self.sequence == '1' :
            checksum = ord('0')
            reply_pkt = RDTReceiver.make_reply_pkt('0',checksum)
         else :
            checksum = ord('1')
            reply_pkt = RDTReceiver.make_reply_pkt(‘1’,checksum)
```

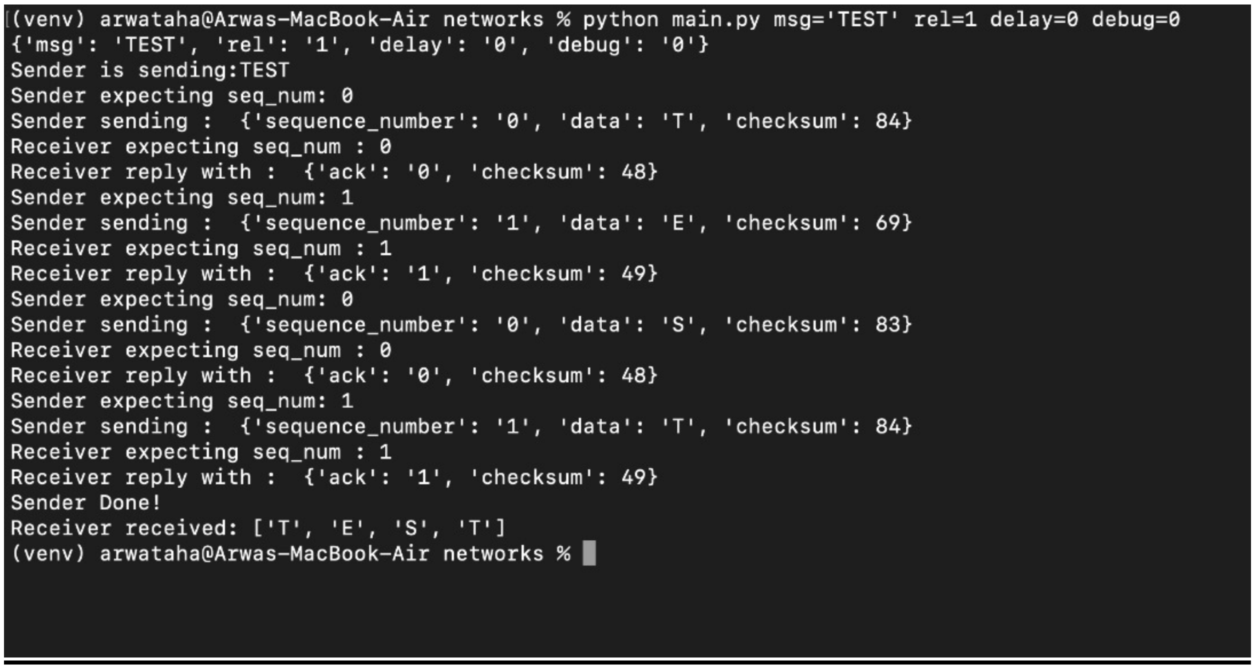
Explanation :

if the received packet is not corrupted and sequence number is same as the expected, we create a reply packet using the *make_reply_pkt() function* using the expected sequence number and its Ascii code, we also send data received

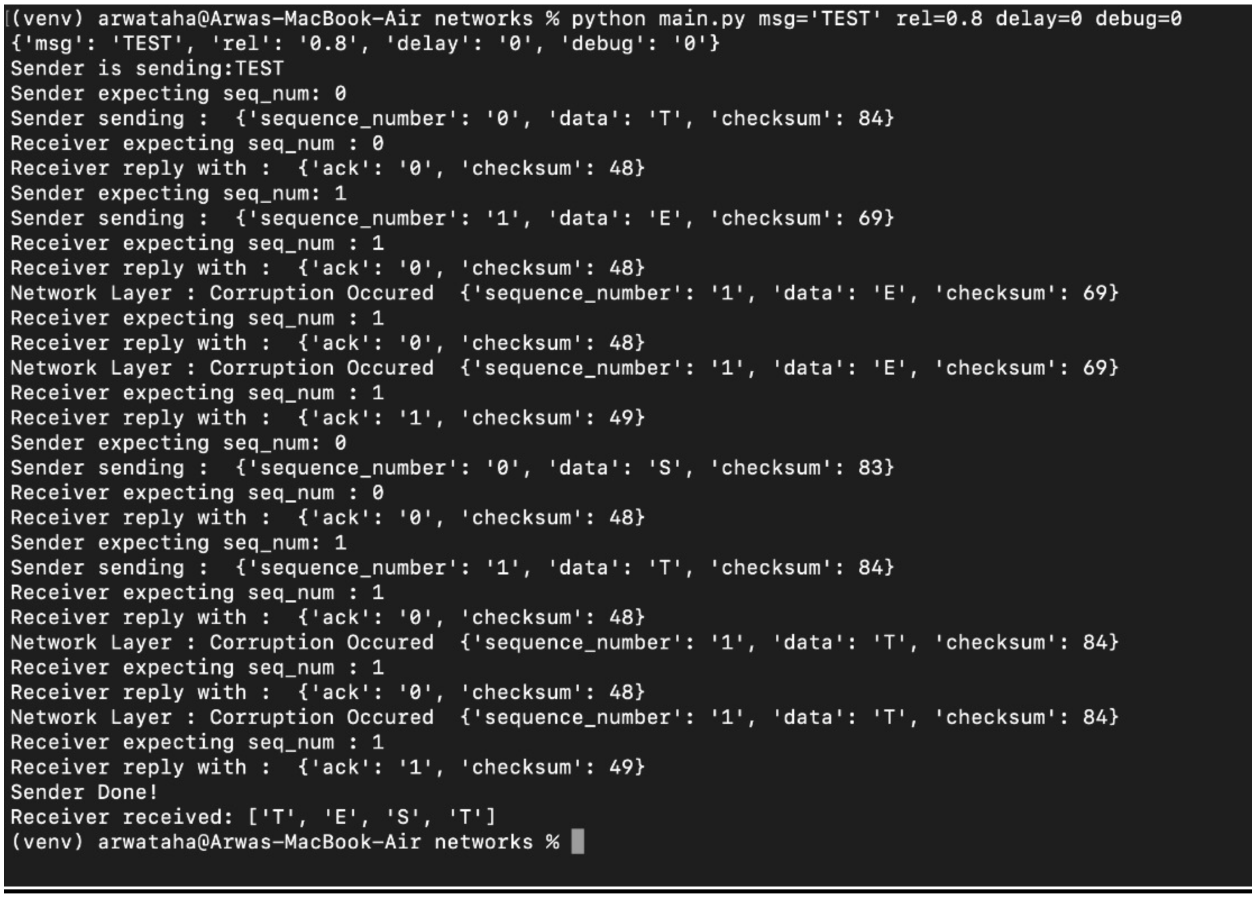
rcv_pkt['data'] using the function *deliver_data(data_recieved)* ,we then alternate sequence number if it's 0 we make it 1 and vice versa .

On the other hand, if the packet received is corrupted or sequence number is not correct we will change the sequence number of the packet and get Ascii code of the new sequence number, then create a new reply packet using the *make_reply_pkt() function*, using the expected sequence number and its ascii code.

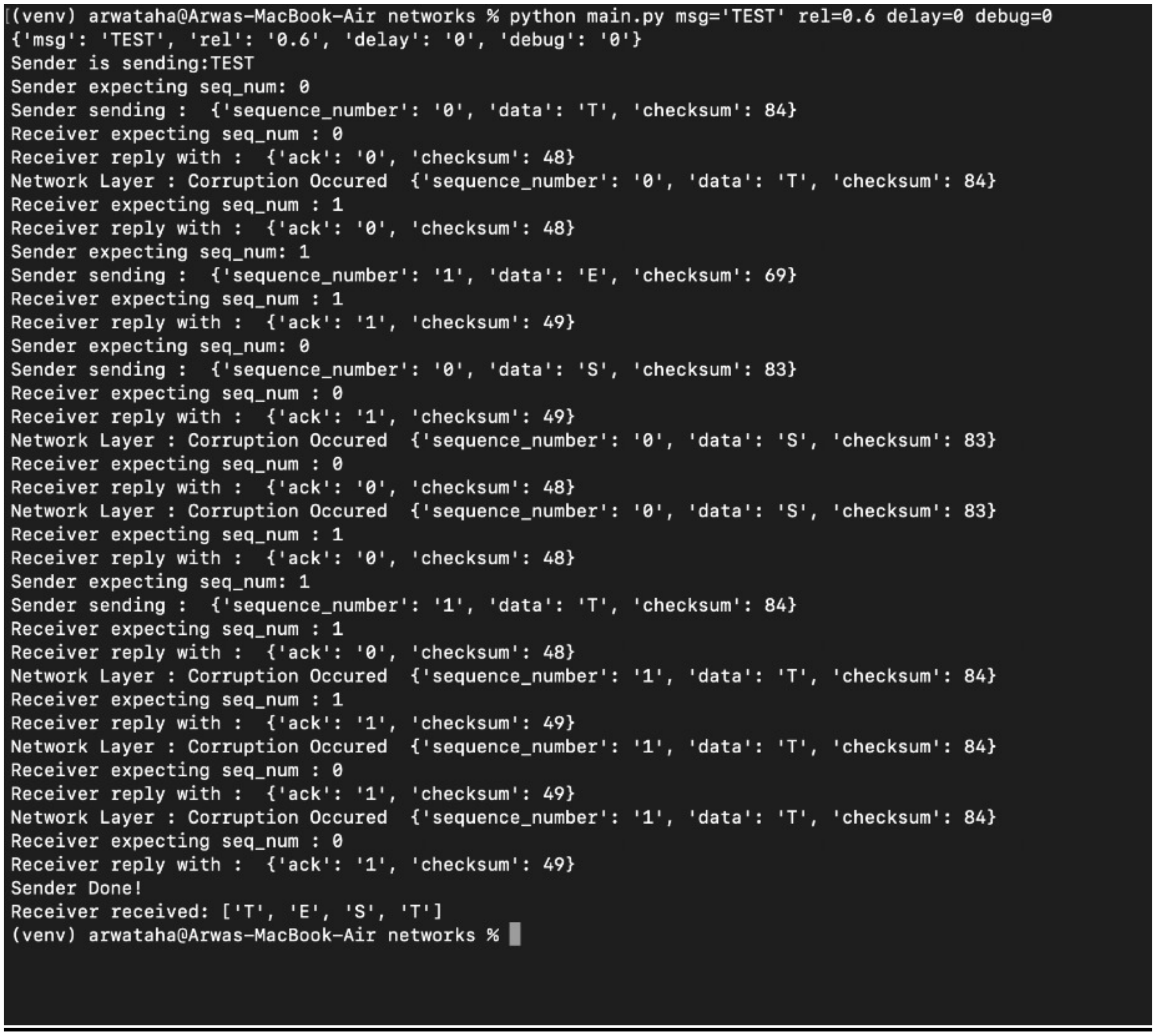
Test cases screenshots:



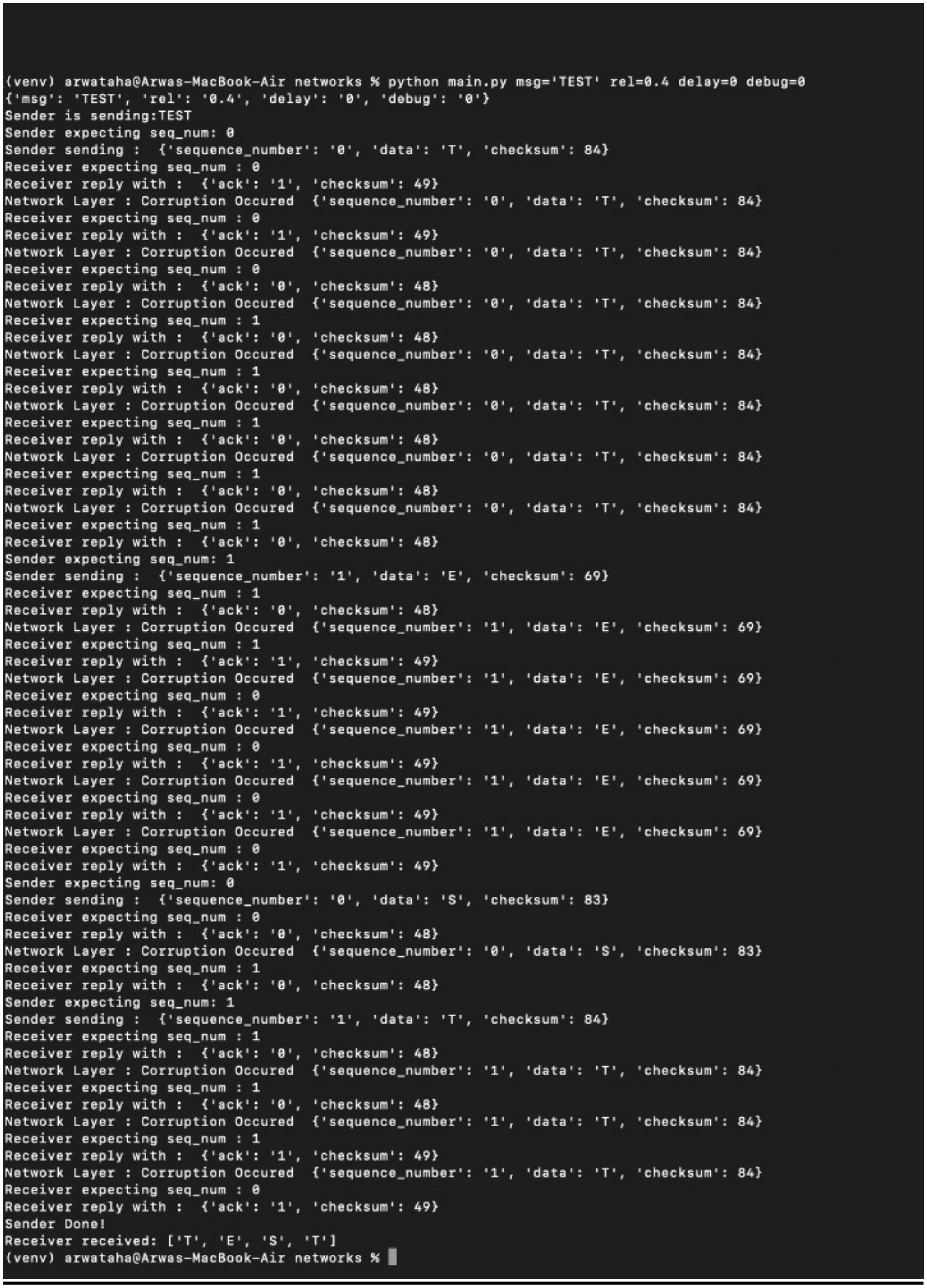
```
(venv) arwataha@Arwas-MacBook-Air networks % python main.py msg='TEST' rel=1 delay=0 debug=0
{'msg': 'TEST', 'rel': '1', 'delay': '0', 'debug': '0'}
Sender is sending:TEST
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender Done!
Receiver received: ['T', 'E', 'S', 'T']
(venv) arwataha@Arwas-MacBook-Air networks %
```



```
(venv) arwataha@Arwas-MacBook-Air networks % python main.py msg='TEST' rel=0.8 delay=0 debug=0
{'msg': 'TEST', 'rel': '0.8', 'delay': '0', 'debug': '0'}
Sender is sending:TEST
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender Done!
Receiver received: ['T', 'E', 'S', 'T']
(venv) arwataha@Arwas-MacBook-Air networks %
```



```
(venv) arwataha@Arwas-MacBook-Air networks % python main.py msg='TEST' rel=0.6 delay=0 debug=0
{'msg': 'TEST', 'rel': '0.6', 'delay': '0', 'debug': '0'}
Sender is sending:TEST
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender Done!
Receiver received: ['T', 'E', 'S', 'T']
(venv) arwataha@Arwas-MacBook-Air networks %
```



```
(venv) arwataha@Arwas-MacBook-Air networks % python main.py msg='TEST' rel=0.4 delay=0 debug=0
{'msg': 'TEST', 'rel': '0.4', 'delay': '0', 'debug': '0'}
Sender is sending:TEST
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'E', 'checksum': 69}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender expecting seq_num: 0
Sender sending :  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '0', 'data': 'S', 'checksum': 83}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Sender expecting seq_num: 1
Sender sending :  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '0', 'checksum': 48}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 1
Receiver reply with :  {'ack': '1', 'checksum': 49}
Network Layer : Corruption Occured  {'sequence_number': '1', 'data': 'T', 'checksum': 84}
Receiver expecting seq_num : 0
Receiver reply with :  {'ack': '1', 'checksum': 49}
Sender Done!
Receiver received: ['T', 'E', 'S', 'T']
(venv) arwataha@Arwas-MacBook-Air networks %
```