```python
import socket
from random import random
from time import sleep
import struct

# Global Variables
host = None
port = None


# UDP Checksum Function
def checksum_func(data):
    checksum = 0
    data_len = len(data)

    # Appends 0's to the end of data and adjusts data_len
    if (data_len % 2):
        data_len += 1
        data += struct.pack('!B', 0)

    # Compute the sum
    for i in range(0, data_len, 2):
        w = (data[i] << 8) + (data[i + 1])
        checksum += w

    # Wrap around bit
    checksum = (checksum >> 16) + (checksum & 0xFFFF)

    # Complement the result
    checksum = ~checksum & 0xFFFF
    return checksum


# Create Socket
def socket_create():
    global host
    global port
    global s
    host = ''
    port = 1001

    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    except socket.error as msg:
        print("Socket creation error: " + str(msg))


# Bind to Socket
def socket_bind():
    global host
    global port
    global s

    try:
        s.bind((host, port))
        print("The server is ready to receive")
    except socket.error as msg:
        print("Socket biding error: " + str(msg))


# State 0
def state0():
    global s

    try:
```

```
67          data, addr = s.recvfrom(2048)
68          print("IP: " + addr[0] + " | Port: " + str(addr[1]))
69          print("Message: " + str(data.decode('utf-8')))
70
71          # Split data to get message and checksum
72          str_data = str(data.decode('utf-8'))
73          message, rcv_checksum = str_data.split("|")
74
75          # Compute checksum
76          # Random is used to simulating data being corrupted...
77          if random() > 0.5:
78              checksum = checksum_func(bytes(message.encode('utf-8')))
79          else:
80              checksum = 0
81
82          if str(checksum) == rcv_checksum:
83              print("Send ACK")
84              s.sendto("ACK".encode('utf-8'), addr)
85
86              # Send message to the application layer
87              print("Message sent to the application layer.")
88
89          else:
90              print("Send NACK")
91              s.sendto("NACK".encode('utf-8'), addr)
92
93      except socket.error as msg:
94          print(str(msg))
95
96      return state0
97
98
99  # Main Function
100 if __name__ == "__main__":
101     global s
102
103     # created socket
104     socket_create()
105     # bind socket
106     socket_bind()
107
108     # Initial State
109     state = state0
110     while state:
111         state = state()
112
113     s.close()
114     print("FSM Done")
115
```