

# **Reliable UDP echo server**

Ahmed Hosni Kmal 15P6023

Gina Jemy Georgui 15P6018

Mahmoud Mohamed Nagy 15P5011

Mohamed Maged 14P8092

# Protocol:

In order to make our UDP echo server connection more reliable and stateful like TCP, we had to implement some features in our application layer.

Features:

* Sequence number for each message that flips every successful acknowledge, to avoid message loss or duplicates.
* Timeout to re-send the message that wasn’t acknowledged by the server or the client.
* Logs printed on the server’s console.

# Client code:

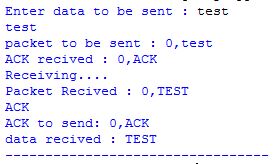
**import** socket  
  
serverName = **'localhost'**serverPort = 12000  
clientSocket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
clientSocket.settimeout(10)  
sentAck = **"0,ACK"**clientSocket.settimeout(0.2)  
  
seqSnd = 0  
seqRcvd = 0  
  
  
  
  
  
**def** isACK(packet):  
 packet = packet.decode(**"UTF-8"**)  
 arr = packet.split(**","**)  
 **if** arr[1] == **"ACK" and** int(arr[0]) == seqSnd:  
 **return** 1  
 **else**:  
 **return** 0  
  
  
**def** verifySeq(packet):  
 **global** seqRcvd  
 packet = packet.decode(**"UTF-8"**)  
 arr = packet.split(**","**)  
 **if** int(arr[0]) == seqRcvd:  
 **return** 1  
 **else**:  
 **return** 0  
  
  
**def** receive():  
 **global** sentAck  
 **global** seqRcvd  
 data, clientAddress = clientSocket.recvfrom(2048)  
 print(**"Packet Recived : "** + data.decode(**'UTF-8'**))  
 **if** verifySeq(data):  
 sentAck = createPacket(**"ACK"**, seqRcvd)  
 print(**"ACK to send: "** + sentAck)  
 clientSocket.sendto(sentAck.encode(**'UTF-8'**), (serverName, serverPort))  
 data = data.decode(**"UTF-8"**)  
 arr = data.split(**","**)  
 seqRcvd = 1 - seqRcvd  
 **return** arr[1]  
 clientSocket.sendto(sentAck.encode(**'UTF-8'**), (serverName, serverPort))  
 **return** receive()  
  
  
**def** sendPacket(packet):  
 **global** seqSnd  
 **try**:  
 print(**"packet to be sent : "** + packet)  
 clientSocket.sendto(packet.encode(**'UTF-8'**), (serverName, serverPort))  
 ack = clientSocket.recv(2048)  
 print(**"ACK recived : "** + ack.decode(**"UTF-8"**))  
 **if not** (isACK(ack)):  
 ack = clientSocket.recv(2048)  
 **else**:  
 seqSnd = 1 - seqSnd  
 **except** socket.timeout:  
 sendPacket(packet)  
  
  
z  
**def** createPacket(data, seq):  
 print(data)  
  
 packet = str(seq) + **','** + str(data)  
 **return** packet  
  
  
**while** 1:  
 **try**:  
  
 k = input(**"Enter data to be sent : "**)  
 packet\_to\_send = createPacket(k, seqSnd)  
 sendPacket(packet\_to\_send)  
 print(**"Receiving...."**)  
 k = receive()  
 print(**"data recived : "** + k)  
 print(**"---------------------------------"**)  
 **except** socket.error **as** exc:  
 print(**"Server isn't working"**)

# Server code:

ack = **"ACK"**sentAck = **"0,ACK"**serverPort = 12000  
serverSocket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
serverSocket.bind((**''**, serverPort))  
print(**"the server is ready "**)  
seqSnd = 0  
seqRcvd = 0  
**while** 1:  
 **try**:  
 serverSocket.settimeout(0.2)  
  
  
 **def** isACK(packet):  
 **global** seqSnd  
 packet = packet.decode(**"UTF-8"**)  
 arr = packet.split(**","**)  
 **if** arr[1] == **"ACK" and** int(arr[0]) == seqSnd:  
 **return** 1  
 **else**:  
 **return** 0  
  
  
 **def** verifySeq(packet):  
 **global** seqRcvd  
 packet = packet.decode(**"UTF-8"**)  
 arr = packet.split(**","**)  
 **if** int(arr[0]) == seqRcvd:  
 **return** 1  
 **else**:  
 **return** 0  
  
  
 **def** receive():  
 **global** sentAck  
 **global** seqRcvd  
 data, clientAddress = serverSocket.recvfrom(2048)  
 print(**"packet received: "** + data.decode(**"UTF-8"**))  
 **if** verifySeq(data):  
 sentAck = createPacket(ack, seqRcvd)  
 print(**"ACK Packet to send : "** + sentAck)  
 serverSocket.sendto(sentAck.encode(**'UTF-8'**), clientAddress)  
 data = data.decode(**"UTF-8"**)  
 arr = data.split(**","**)  
 seqRcvd = 1 - seqRcvd  
 **return** arr[1], clientAddress  
 serverSocket.sendto(sentAck.encode(**'UTF-8'**), clientAddress)  
 **return** receive()  
  
  
 **def** sendPacket(packet, clientAddress):  
 **global** seqSnd  
 **try**:  
 print(**"Packet to send as a response : "** + packet)  
 serverSocket.sendto(packet.encode(**'UTF-8'**), clientAddress)  
 ack = serverSocket.recv(2048)  
 print(**"ACK received: "** + ack.decode(**'UTF-8'**))  
 **if not** (isACK(ack)):  
 ack = serverSocket.recv(2048)  
 **else**:  
 seqSnd = 1 - seqSnd  
 **except** socket.timeout:  
 sendPacket(packet, clientAddress)  
  
  
 **def** createPacket(data, seq):  
  
 packet = str(seq) + **','** + str(data)  
 **return** packet  
  
  
 **while** 1:  
 k, clientAddress = receive()  
 print(**"data from the packet : "** + k)  
 packet\_to\_send = createPacket(k.upper(), seqSnd)  
 sendPacket(packet\_to\_send, clientAddress)  
 print(**"---------------------------------"**)  
 **except** socket.timeout:  
 x = 1

# Sample run

Client screenshot:



Server screenshot:

