

# Reliable UDP echo server

Ahmed Hosni Kmal 15P6023

Gina Jemy Georgui 15P6018

Mahmoud Mohamed Nagy 15P5011

Mohamed Maged 14P8092

## Protocol:

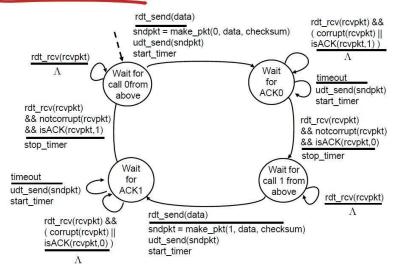
In order to make our UDP echo server connection more reliable and stateful like TCP, we had to implement some features in our application layer.

#### Features:

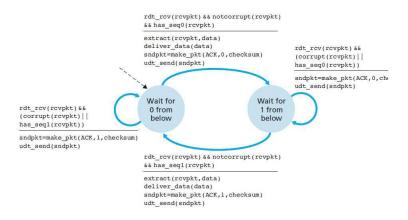
- Sequence number for each message that flips every successful acknowledge, to avoid message loss or duplicates.
- Timeout to re-send the message that wasn't acknowledged by the server or the client.
- Logs printed on the server's console.

State diagram we used

## rdt3.0 sender



Transport Layer 3-39



## 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)
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:

```
Run: dudbsrv dudbclnt
       "C:\Users\Gina Salib\AppData\Local\Programs\Python\Python36-32\pyt
       Enter data to be sent : first message
       first message
packet to be sent : 0, first message
   ACK recived : 0,ACK
       Receiving....
   a
S
       Packet Recived : 0,FIRST MESSAGE
   aCK
       ACK to send: 0,ACK
       data recived : FIRST MESSAGE
       Enter data to be sent : second message
       second message
       packet to be sent : 1, second message
       ACK recived : 1,ACK
       Receiving....
       Packet Recived : 1, SECOND MESSAGE
       ACK to send: 1,ACK
       data recived : SECOND MESSAGE
       Enter data to be sent : third message
       third message
       packet to be sent : 0, third message
       ACK recived : 0,ACK
       Receiving....
       Packet Recived : 0, THIRD MESSAGE
       ACK to send: 0,ACK
        data recived : THIRD MESSAGE
```

#### Server screenshot:

```
Run: udbsrv udbclnt
        "C:\Users\Gina Salib\AppData\Local\Programs\Python\Pythor
( 1
       the server is ready
       packet received: 0, first message
ACK Packet to send : 0,ACK
   data from the packet : first message
    Packet to send as a response : 0, FIRST MESSAGE
S
       ACK received: 0,ACK
×
        packet received: 1, second message
        ACK Packet to send : 1,ACK
        data from the packet : second message
        Packet to send as a response : 1, SECOND MESSAGE
        ACK received: 1,ACK
        packet received: 0, third message
       ACK Packet to send : 0,ACK
       data from the packet : third message
       Packet to send as a response : 0, THIRD MESSAGE
       ACK received: 0,ACK
```