LAB-07

Analyze UDP Packet

Three bytes of data was captured being sent over UDP in this packet printout: L09-UDP-printout.

Drawing

Highlight the source port, destination port, length, and checksum values in the BYTES section of the printout with different colors. Include a legend describing what each color matches with.

```
No.     Leftover Capture Data Time             Source               Destination          Info
Protocol Length Data        Data
     55                        11.190999       10.31.12.33          93.184.216.34        51081 → 1234 Len=3
UDP      45                 68690a
Frame 55: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface en0, id 0
Ethernet II, Src: Apple_31:3c:75 (8c:85:90:31:3c:75), Dst: Alcatel-_f2:8e:01 (e8:e7:32:f2:8e:01)
Internet Protocol Version 4, Src: 10.31.12.33, Dst: 93.184.216.34
User Datagram Protocol, Src Port: 51081, Dst Port: 1234
Data (3 bytes)
0000  e8 e7 32 f2 8e 01 8c 85 90 31 3c 75 08 00 45 00   ..2......1<u..E.
0010  00 1f d1 91 00 00 40 11 5d 22 0a 1f 0c 21 5d b8   ......@.]"...!].
0020  d8 22 c7 89 04 d2 00 0b 74 f8 68 69 0a            ."......t.hi.
```

Source port: 0xC789

Dest Port: 0x04d2

Length: 0x000b

Checksum: 0x74f8

Data: 0x68690a

RDT qanda

part I

Answer the following questions with one of these answers:

retransmission	**ACK**	**sequence numbers**	**NAK**	**checksum**

1. **Let's the sender know that a packet was NOT received correctly at the receiver.**

Ans. NAK

2. **Used by sender or receiver to detect bits flipped during a packet's transmission.**

Ans. Retransmission

3. **Allows for duplicate detection at receiver.**

Ans. Sequence number

4. **Let's the sender know that a packet was received correctly at the receiver.**

Ans. ACK

5. Allows the receiver to eventually receive a packet that was corrupted or lost in an earlier transmission.

Ans. Checksum

qanda part II

1. Looking at RDT 2.0 shown below, make a list of valid possible sequences of transitions after S1. (S1 → …, S1 → …, etc.)

Ans. S1->R1->S2->R1->S2…

2. Looking at RDT 2.1, what state is the sender in after having sent data with a sequence number of 1.

Ans. Waiting for ACK or NAK.

3. What is the purpose of the sliding windows?

Ans. Single mechanism that supports:

-Multiple outstanding packets

- Reliable delivery

- In-order delivery

- Flow control

Sender and receiver each maintain "window" abstractions to track outstanding packets Go-Back-N is a special case

-Receive window size of one

4. **What are the four categories for packets within the sliding window?**

Ans. During the transmission process, the data packets pass through one of four stages:

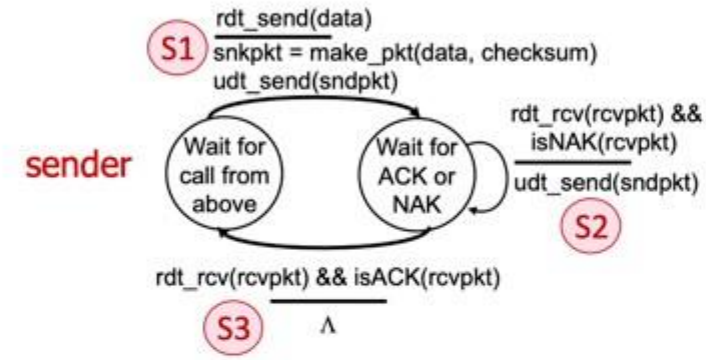
- Sent and acknowledged by the receiver.
- Sent but not acknowledged by the receiver.
- Not sent but the receiver is ready accept them.
- Not sent and the receiver is not ready to accept them.

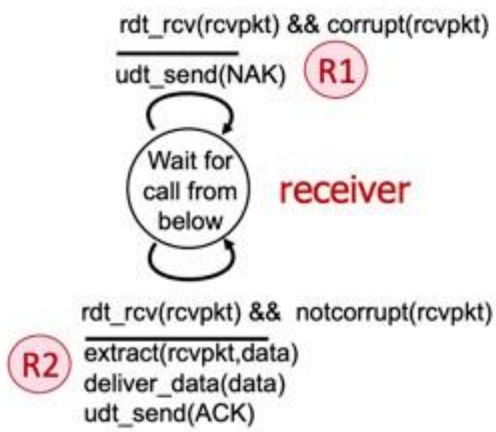
5. **When does the sliding window "slide?"**

Ans. The sender picks a window size, winsize. The basic idea of sliding windows is that the sender is allowed to send this many packets before waiting for an ACK. More specifically, the sender keeps a state variable last_ACKed**,** representing the last packet for which it has received an ACK from the other end; if data packets are numbered starting from 1 then initially last_ACKed = 0. At any instant, the sender may send packets numbered last_ACKed + 1 through last_ACKed + winsize; this packet range is known as the window. Generally, if the first link in the path is not the slowest one, the sender will most of the time have sent all these.

If ACK[N] arrives with N > last_ACKed (typically N = last_ACKed+1), then the window *slides forward*; we set last_ACKed = N. This also increments the upper edge of the window, and frees the sender to send more packets.

RDT 2.0


S1
rdt_send(data)
snkpkt = make_pkt(data, checksum)
udt_send(sndpkt)
sender
Wait for call from above
Wait for ACK or NAK
rdt_rcv(rcvpkt) && isNAK(rcvpkt)
udt_send(sndpkt)
S2
rdt_rcv(rcvpkt) && isACK(rcvpkt)
Λ
S3


rdt_rcv(rcvpkt) && corrupt(rcvpkt)
udt_send(NAK)
R1
Wait for call from below
receiver
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
R2
extract(rcvpkt,data)
deliver_data(data)
udt_send(ACK)