

ID:1407870

Name: Muhan Guan

Q1.

1).

$total\ size\ of\ clip = 0.5\ (s) \times 30\ (fps) \times (1280 \times 720)\ (pixels) \times 3\ (bytes/pixel) \times 8 = 331776000\ bit$

$$T_{delay} = \frac{total\ size\ of\ clip}{rate\ of\ transmission} = \frac{331776000\ (bit)}{1 \times 10^6\ (bps)} = 331.776\ sec$$

$$P_{delay} = \frac{length\ of\ channel}{speed\ of\ signals} = \frac{10000(km)}{200000(km/s)} = 0.05\ sec$$

$$Latency = L = T_{delay} + P_{delay} = 331.776(sec) + 0.05(sec) = 331.826\ sec$$

2).

$$T_{delay} = \frac{total\ size\ of\ clip}{rate\ of\ transmission} = \frac{331776000\ (bit)}{100 \times 10^6\ (bps)} = 3.31776\ sec$$

$$P_{delay} = \frac{length\ of\ channel * 2}{speed\ of\ signals} = \frac{2 * 36000(km)}{300000(km/s)} = 0.24\ sec$$

$$Latency = L = T_{delay} + P_{delay} = 3.31776(sec) + 0.24(sec) = 3.55776\ sec$$

Q2.

1).

Given it's a noisy channel, we can use Shannon's theorem:

$$Max.\ data\ rate = B \times \log_2 \left(1 + \frac{S}{N} \right) = 8 \times 10^3 \times \log_2 \left(1 + \frac{S}{N} \right) \geq 64 \times 10^3\ (bps)$$

$$then\ we\ can\ get: \frac{S}{N} \geq 255$$

$$SNR(dB) = 10 \times \log_{10} \left(\frac{S}{N} \right) \geq 10 \times \log_{10}(255)$$

$$10 \times \log_{10}(255) = 24.0654\ dB$$

In conclusion, to support a data rate of 64 kbps, the minimum signal-to-noise ratio (in dB) should be 24.0654 (dB).

2).

Given it's a noiseless channel, we can use Nyquist's theorem:

$$Max.\ data\ rate = 2B \times \log_2 V \geq 64 \times 10^3\ (bps)$$

$$then\ we\ can\ get: V \geq 16$$

In conclusion, we should use the signal whose level is larger than 16 so that we support a data rate of 64kbps.

Q3.

1). 8 fragments

Each 32-bit data block needs to be split into 8 fragments and transmitted using 8 Hamming (7,4) codes blocks.

2).

$$7 * 8 = 56\ bits$$

3).

We can interleave codewords in continuous stripe to prevent the effects of consecutive bit (burst) errors. Specifically, we compose the first bit of each encoded block into a new

interleaved block, the second bit into another interleaved block, and so on until all the bits are organized into a new, interleaved data block, then we can transmit this interleaved data block to make our system robust to sequences of errors.

Q4.

1. Modularization eases maintenance: Each layer in the model performs a specific function, and the layers are designed to be modular, which makes it easier to upgrade or replace specific components of the network without affecting the entire system.
2. Reduced Complexity: Each layer is independent, and each layer provides services up and down through the interlayer interface without exposing the internal implementation. Since each layer only implements a relatively independent function, an intractable complex problem can be decomposed into several smaller problems that are easier to handle, so that the complexity of the whole problem is reduced.
3. Better scalability: Layered models are scalable, which means that additional layers can be added to accommodate new technologies or protocols. This allows the model to adapt to changing requirements without requiring a complete redesign.

Q5.

1).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000	10.13.185.238	posh.austlii.edu.au	TCP	76	64920 → 80 [SYN] Seq=8 Min=65535 Len=0 MSS=1460
2	0.01689	posh.austlii.edu.au	10.13.185.238	TCP	58	80 → 64920 [576, ACK] Seq=8 Ack=1 Win=0 Len=0
3	0.01683	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=1 Ack=1 Min=65535 Len=0
4	0.01663	10.13.185.238	posh.austlii.edu.au	TCP	76	64921 → 80 [SYN] Seq=8 Min=65535 Len=0 MSS=1460
5	0.032152	posh.austlii.edu.au	10.13.185.238	TCP	54	[TCP Window Update] 80 → 64920 [ACK] Seq=1 Ack=1
6	0.032279	10.13.185.238	posh.austlii.edu.au	HTTP	426	GET / HTTP/1.1
7	0.034023	posh.austlii.edu.au	10.13.185.238	TCP	58	80 → 64921 [576, ACK] Seq=8 Ack=1 Win=0 Len=0
8	0.034494	10.13.185.238	posh.austlii.edu.au	TCP	54	64921 → 80 [ACK] Seq=1 Ack=1 Min=65535 Len=0
9	0.050925	posh.austlii.edu.au	10.13.185.238	TCP	54	80 → 64920 [ACK] Seq=1 Ack=373 Win=6800 Len=0
10	0.050987	posh.austlii.edu.au	10.13.185.238	TCP	54	[TCP Window Update] 80 → 64921 [ACK] Seq=1 Ack=1
11	0.082624	posh.austlii.edu.au	10.13.185.238	TCP	1384	80 → 64920 [ACK] Seq=1 Ack=373 Win=6800 Len=12
12	0.082645	posh.austlii.edu.au	10.13.185.238	TCP	1384	80 → 64920 [ACK] Seq=1251 Ack=373 Win=6800 Len=0
13	0.082699	posh.austlii.edu.au	10.13.185.238	TCP	1384	80 → 64920 [ACK] Seq=2581 Ack=373 Win=6800 Len=0
14	0.082787	posh.austlii.edu.au	10.13.185.238	TCP	1384	80 → 64920 [ACK] Seq=3751 Ack=373 Win=6800 Len=0
15	0.082794	posh.austlii.edu.au	10.13.185.238	TCP	1384	80 → 64920 [ACK] Seq=4801 Ack=373 Win=6800 Len=0
16	0.082799	posh.austlii.edu.au	10.13.185.238	TCP	1384	80 → 64920 [ACK] Seq=6251 Ack=373 Win=6800 Len=0
17	0.082802	posh.austlii.edu.au	10.13.185.238	HTTP	1188	HTTP/1.1 200 OK (text/html)
18	0.082823	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=1251 Win=65535 Len=0
19	0.083466	posh.austlii.edu.au	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=2581 Win=65535 Len=0
20	0.083469	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=3751 Win=65535 Len=0
21	0.083468	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=5001 Win=65535 Len=0
22	0.083468	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=6251 Win=65535 Len=0
23	0.083468	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=7281 Win=65535 Len=0
24	0.083469	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=8035 Win=65535 Len=0
25	5.159816	posh.austlii.edu.au	10.13.185.238	TCP	54	80 → 64920 [FIN, ACK] Seq=8635 Ack=373 Win=6800 Len=0
26	5.159903	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [ACK] Seq=373 Ack=8636 Win=65535 Len=0
27	5.159755	10.13.185.238	posh.austlii.edu.au	TCP	54	64920 → 80 [FIN, ACK] Seq=373 Ack=8636 Win=65535 Len=0
28	5.173822	posh.austlii.edu.au	10.13.185.238	TCP	54	80 → 64920 [ACK] Seq=8636 Ack=374 Win=6800 Len=0

IP address of source:10.13.183.179

IP address of destination: 138.25.65.147

website: <http://www.austlii.edu.au/>

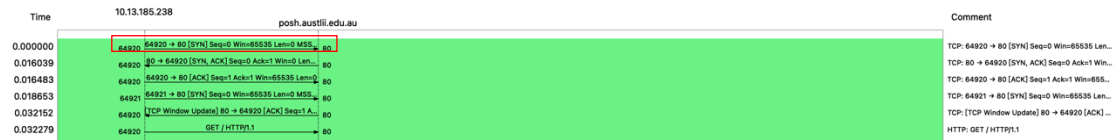
Way of validation: For the IP address of source, we can validate it by the terminal command line (ifconfig en0) to check if it is the same with the 'Source' column.

```
(base) guanmuhandeMacBook-Pro:~ guanmuhan$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 8c:85:90:67:97:4a
inet6 fe80::182b:9744:700d:2f41%en0 prefixlen 64 secured scopeid 0x5
inet 10.13.185.238 netmask 0xffffe000 broadcast 10.13.191.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
```

For the IP address of destination, we can validate it by typing it directly into browser to check if it is the same website we are capturing.

2).

CONNECT(red square): The client calls the CONNECT primitive to send a SYN packet to the server, indicating a request to establish a TCP connection.



DISONNECT(black square): The server calls the DISCONNECT primitive to send FIN and ACK packets to client ,indicating a request to terminate a connection.

