Homework 1

1.

a) delay sensitive - Opening a file involves less message transfers.
b) bandwidth sensitive – Many message transfer is required for reading a file.
c) delay sensitive – Contents of a directory are of small size.
d) delay sensitive – Attributes of a files are smaller size than the actual file itself.
e) bandwidth sensitive – Average video file is larger than common files.

2.

a) Transmit Time for single link = Size / Bandwidth
= 10000 / 100 Mbps
= 100 μs
Propagation delay for each link = 20 μs
With one switch, there will be two links,
Therefore, required total latency = 2 * Transmit Time + 2 * Propagation Delay
= 2 * 100 + 2 * 20 μs
= **240 μs**

b) With three switches, there will be four links,
So, required total latency = 4 * Transmit Time + 4 * Propagation Delay
= 4 * 100 + 4 * 20 μs
= **480 μs**

c) With cut-through switching, the switch delay = 200 / 100 Mbps = 2 μs
Therefore, required total latency = Transmit time + Propagation Delay + Switch delay
= 100 + 20 + 2
= **122 μs**

3.
Bandwidth between A and S = 100 Mbps
Latency between A and S = 20 μs
Bandwidth between B and S = 50 Mbps
Latency between B and S = 30 μs
Switch delay = 40 μs
a)
Transmit Time A to S= Size / Bandwidth
= 10000 / 100 Mbps
= 100 μs
Time needed to send 10,000 bit from A to S = Transmit Time + Propagation Delay
= 100 + 20
= 120 μs

Transmit Time S to B = Size / Bandwidth
= 10000 / 50 Mbps
= 200 μs
Time needed to send 10,000 bit from S to A = Transmit Time + Propagation Delay
= 200 + 30
= 230 μs
Total time taken to send 10,000 bit packet from A to B = Total time from A to S + Total time from S to B + switch delay
= 120 + 230 + 40
= **390 μs.**

b) Bandwidth between A and S is 100 Mbps and between S and B is 50 Mbps, So, 50 Megabits should be buffered for every second. So S has to provide buffering of **5Mb** such that packets won't be dropped until after 100ms has elapsed.

c) Total time taken to transmit 10,000 bit from A to B = 390 μs.
Time taken to transmit signal from B to A = Switch delay + propagation delay
= 40 μs + 50 μs
= 90 μs
Total RTT between A and B = 390 + 90 = 480 μs
Total bits sent by A during this RTT = 480 * 10 ^ -6 * 100 * 10 ^ 6
= 48000 bits
So approximate number of packets sent by A is **5**.

4. Packet overhead = 50 bytes
Total number of bytes of data sent = 1 million bytes
= 1 * 10 ^ 6 bytes

<u>Packet size – 1000 data bytes</u>
Total number of packets = 1 * 10 ^ 6 / 1000
= 1000 packets
Total overhead = 1000 * 50 = 50000 bytes
Loss in bytes = 1000 bytes (Since overhead is added in total overhead)
Total overhead + loss for 1000 data bytes = 51000 bytes

<u>Packet size – 10000 data bytes</u>
Total number of packets = 1 * 10 ^ 6 / 10000
= 100 packets
Total overhead = 100 * 50 = 5000 bytes
Loss in bytes = 10000 bytes (Since overhead is added in total overhead)
Total overhead + loss for 10000 data bytes = 15000 bytes

Packet size – 20000 data bytes

Total number of packets = 1 * 10 ^ 6 / 20000

= 50 packets

Total overhead = 50* 50 = 2500 bytes

Loss in bytes = 20000 bytes (Since overhead is added in total overhead)

Total overhead + loss for 20000 data bytes = 22500 bytes

The total number of overhead + loss byte for packet data sizes of

1000 = 51000 bytes

10000 = 15000 bytes

20000 = 22500 bytes

By overhead and loss,

Packet data size of **10000 is optimal**.

5)

a)

```
[hacedor91:proj1 lovelinanand$ ping www.syracuse.edu
PING syracuse.edu (128.230.18.198): 56 data bytes
64 bytes from 128.230.18.198: icmp_seq=0 ttl=51 time=40.864 ms
64 bytes from 128.230.18.198: icmp_seq=1 ttl=51 time=50.849 ms
64 bytes from 128.230.18.198: icmp_seq=2 ttl=51 time=42.532 ms
64 bytes from 128.230.18.198: icmp_seq=3 ttl=51 time=44.663 ms
64 bytes from 128.230.18.198: icmp_seq=4 ttl=51 time=42.749 ms
64 bytes from 128.230.18.198: icmp_seq=5 ttl=51 time=129.441 ms
64 bytes from 128.230.18.198: icmp_seq=6 ttl=51 time=232.322 ms
^C
--- syracuse.edu ping statistics ---
7 packets transmitted, 7 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 40.864/83.346/232.322/67.601 ms
```

Ping response from www.syracuse.edu:

Round trip minimum = 40.864 ms

Round trip maximum = 232.322 ms

Round trip average = 40.864 ms

```
PING www.cam.ac.uk (131.111.150.25): 56 data bytes
64 bytes from 131.111.150.25: icmp_seq=0 ttl=39 time=750.848 ms
64 bytes from 131.111.150.25: icmp_seq=1 ttl=39 time=400.096 ms
64 bytes from 131.111.150.25: icmp_seq=2 ttl=39 time=426.543 ms
Request timeout for icmp_seq 3
Request timeout for icmp_seq 4
64 bytes from 131.111.150.25: icmp_seq=3 ttl=39 time=2021.829 ms
64 bytes from 131.111.150.25: icmp_seq=4 ttl=39 time=1085.643 ms
64 bytes from 131.111.150.25: icmp_seq=5 ttl=39 time=747.365 ms
64 bytes from 131.111.150.25: icmp_seq=6 ttl=39 time=288.735 ms
^C
--- www.cam.ac.uk ping statistics ---
7 packets transmitted, 7 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 288.735/817.294/2021.829/552.699 ms
```

Ping response from www.cam.ac.uk:

Round trip minimum = 288.735 ms
Round trip maximum = 2021.829 ms
Round trip average = 817.294 ms

b) From the result, we can infer that RTT from www.syracuse.edu is faster than www.cam.ac.uk

Possible reasons for the differences are

1) Server location and distance from Binghamton.
2) Number of hops between server and the client.
3) Traffic load in these servers.

6)

1 0 1 0	0
1 0 1 0	0
1 1 0 0	0
1 1 0 1	1
0 1 0 1	1

So the data to be sent is **1010 0 1010 0 1100 0 1101 1 0101 1**

7)

a) Data = 1011 0010 011

Polynomial = 10011 (x^4 + x + 1)

10011 | 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0

1 0 0 1 1

0 0 1 0 1 0 1

1 0 0 1 1

0 0 1 1 0 0 0

1 0 0 1 1

0 1 0 1 1 1

1 0 0 1 1

0 0 1 0 0 1 0

1 0 0 1 1

0 0 0 0 **1 0 0 0 - CRC**

Final data to be transmitted is:

1011 0010 0111 000

b)

Error pattern = 1000 1000 0000 000

Data received with error = 0011 1010 0111 000

10011 | 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0

1 0 0 1 1

1 0 1 0 0

1 0 0 1 1

0 0 1 1 1 0 1

1 0 0 1 1

0 1 1 1 0 0

1 0 0 1 1

0 1 1 1 1 0

1 0 0 1 1

0 1 1 0 1 1

1 0 0 1 1

0 1 0 0 0 1

1 0 0 1 1

0 0 0 1 0 1 0 0

1 0 0 1 1

0 0 1 1 1 0

Since the remainder is not zero, this shows the input data has error. So error can be detected.

c)

Error pattern = 1001 1000 0000 000

Data received with error = 0010 1010 011 000

```
10011 | 0 0 1 0 1 0 1 0 0 1 1 0 0 0
        1 0 0 1 1
        1 0 1 1 0
        1 0 0 1 1
        0 0 1 0 1 0 1
            1 0 0 1 1
            0 0 1 1 0 0 0
                1 0 0 1 1
                 0 1 0 1 1 1
                   1 0 0 1 1
                   0 0 1 0 0 1 0
                       1 0 0 1 1
                       0 0 0 0 1 0 0
```

Since the remainder is **not zero**, the error can be detected in the received data

8)

Input = 1010 1010 101

a) Minimum number of parity bits required is 4 (_ _ 1 _ 0 1 0 _ 1 0 1 0 1 0 1)

b) Hamming code for the above message is

$\underline{1}$ $\underline{0}$ 1 $\underline{1}$ 0 1 0 $\underline{0}$ 1 0 1 0 1 0 1