S-72.3340 Optical Networks

Exercise 1 Solutions

1) In an experiment designed to measure the attenuation coefficient of an optical fiber, the output power from optical source is coupled onto a length of the fiber and measured at the other end. If a 10-km long spool of fiber is used the received optical power is -20 dBm. Under identical conditions but with a 20-km long spool of fiber (instead of the 10-km long spool), the received optical power is -23 dBm. What is the value of the fiber attenuation coefficient (in dB/km)?

1. To measure attenuation coefficient α of fiber

$$L_1 = 10\,km \Longrightarrow P_{r1} = -20\,dBm$$
$$L_2 = 20\,km \Longrightarrow P_{r2} = -23\,dBm$$

$$\begin{aligned}\alpha &= \frac{|additional\,loss|}{added\,length} \\ &= \frac{P_{r2} - P_{r1}}{L_2 - L_1} \\ &= \frac{3dB}{10km} \\ &= 0.3dB/km\end{aligned}$$

2) A very simple line code used in early optical data networks is called *bit stuffing*. The objective of this code is to prevent long runs of 1s and 0s but not necessarily to achieve DC balance. The encoding works as follows. Suppose the maximum number of consecutive 1s that are allowed in the bit stream is k. Then the encoder inserts a 0 bit whenever it sees k consecutive 1 bits in the input sequence.

a) Suppose the incoming data to be transmitted is 1111111111001000000 (read left to right). What is the encoded bit stream, assuming $k = 5$?

Now since k=5 then the bit stuffing encoded bit is:

11111<u>0</u>11111<u>0</u>10010000010

Note that the stuffed bits are underlined above.

b) What is the algorithm used by the decoder to recover the data? Suppose the received bit stream is 01111101011111100011 (read left to right). What is the decoded bit stream?

The decoded bit stream is:

01111110111110011

Several ways possible to represent the decoder algorithm. An example pseudo-code is used here.

```
Decoding Algorithm()
{
  set "1" bit counter to 0;
  count "1" bits in incoming k bit block ;
  if(counter == k)
    {
      output the bits in the block;
      delete the "0" bit after the block;
    }
  /* Time to exit?  */
  if(if next incoming block < k bits long)
   {
      output the bits in the block;
      break;
   }
}
```