Midterm Exam - Fall 2010

Problem (1)

(1) provide an obstraction for layer implementation we can change the protocol of one layer without changing protocols in other layer

(2) we can change the services offered by a layer without changing the protocols used between peers in that layer

Problem (2)

(a) Physical layer
(b) Transport layer (layer 4)

Problem (3)

At the Physical layer the size of the Packet $\sum_{i=1}^{n} h_i + M$

The number of Packets sent Per Second

$$= \frac{\overset{\text{physical capacity (bandwidth)}}{C}}{\sum_{i=1}^{n} h_i + M}$$

=> The band width as seen by app. $A = \#$ of messages Per second x message size

$$\Rightarrow \boxed{A = \frac{C}{\sum_{i=1}^{n} h_i + M} \times M}$$ ↳ get C

Problem(4) Provide a real life example for each

a) Connection oriented : TCP, PPP, HDLC

b) DL layer : Ethernet CSMA / $\underset{CD}{CA}$, PPP, HDLC

c) Connectionless : UDP , IP v4/v6 , Ethernet

d) Error detection : Parity, CRC

e) Point to multi point : Ethernet

f) Transport layer : TCP, UDP

Problem (5)

a) $\therefore$ $2d + 1 = 5$ ⇒ minimum hamming distance $= 5$

correct $\overset{\downarrow}{2}$ errors

b) I can't use $m+r+1 \leq 2^r$ because it is for 1 error

For each valid codeword, we have to reserve 1 codeword for the CW itself

+ n invalid codewords such that the distance between each invalid CW and valid CW is "1"

$\frac{n(n-1)}{2}$ ↖

+ ${}^{n}C_2$ invalid CWs such that distance between each invalid CW and valid CW is "2"

$\Rightarrow$ For every valid CW we have to reserve

$$\left[1 + n + \frac{n(n-1)}{2} \right]$$

$\because$ we have 2^m valid code words

$\Rightarrow$ The minimum no. of code words $= \boxed{2^m \left[1 + n + \frac{n(n-1)}{2} \right]}$

Problem(6)

(a) No modification (I am not interested in reverse channel)

(b) $\because$ There is loss I need

(1) ACK

(2) time out $\Big\} \Rightarrow$ due to loss / corruption

(3) $\because$ Propagation delay > 0

$\Rightarrow$ duplication can happen

I need sequence number.

Protocol

Sender:

```
  nf = o
from_NL( & buffer)
while (true) {
    S.info = buffer
    S.seq = nf
   To_Physical_layer (S)
   start_timer (s.seq)
   wait_for_event(event)
if (event = = from arrival) {
  if( r.ack = = nf ) {
     stop_timer( )
     nf = nf + 1
     from_network_layer (buffer)
  }
 }
}
```

receiver:

```
 fe = o
 while ( true ) {
 wait_for_event(event)
if(event = = frame_arrival) {
from_Physical_layer (r)
if(r.seq = = fe )
to_network_layer ( )
 fe = fe + 1
r.ack = fe - 1
to_Physical_layer (r)
  }
}
```

I don't need these statements because the channel from Rx → Tx is ideal so I need any acks from Rx → Tx and Tx needn't check on ACK on what frame (i.e. Sender will always receive an Ack

(c) No modification
(sender is free to send @ any time)
even if buffer is full, I will wait)

(d) ∵ only ambiguity @ the receiver
⇒ I need sequence ⇒ 1 bit

I don't need ACK

Protocol

```
sender
  NF = 0
while (true) {
from_NL(& buffer)
  S.seq = nf
  S.info = buffer
 to_physical layer(s)
nf = nf + 1
}
```

```
receiver
fe = 0
while (true) {
wait_for_event(event)
from_physical_layer (r)
if r.seq = fe {
to_network_layer
  fe = fe + 1 }
}
```

(e) No modification