Solution to HW5

11.2 (15 pts.)

a. A total of 2i bits are in incorrect (i bits delivered to the wrong destination + i bits not delivered to the correct destination). Thus

$$B1 = \left(\frac{h}{h+i} \times B \times 2i\right) + \left(\frac{i}{h+i} \times B\right) = \frac{i(2h+1)}{h+i} \times B$$

$$M1 = \frac{B1}{B} = \frac{i(2h+1)}{h+i} = \frac{2h+1}{1+h/i}$$

b. In this case we are only concerned with the i bits not delivered to the correct destination; because the error is detected, the cell will not be delivered to the wrong destination.

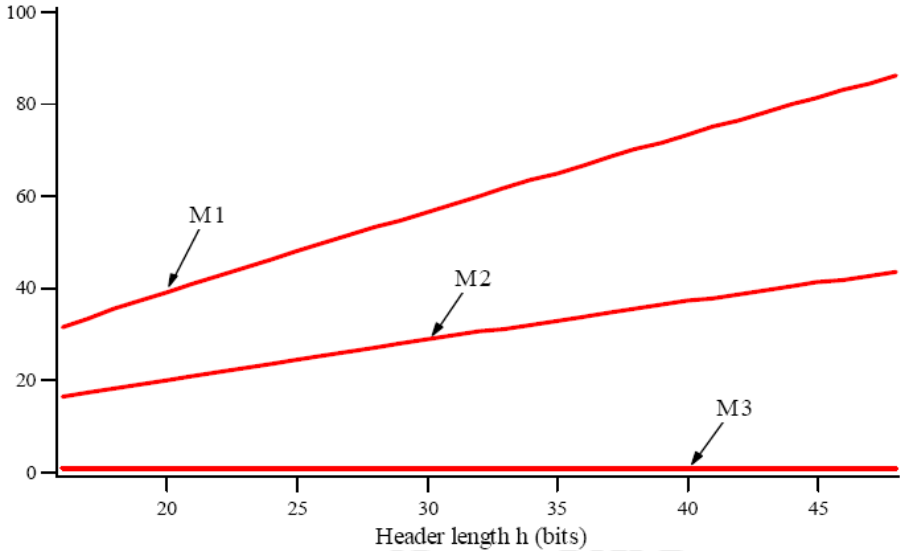
$$B2 = \left(\frac{h}{h+i} \times B \times i\right) + \left(\frac{i}{h+i} \times B\right) = \frac{i(h+1)}{h+i} \times B$$

$$M2 = \frac{B2}{B} = \frac{i(h+1)}{h+i} = \frac{h+1}{1+h/i}$$

c. Because the header errors are corrected, all information bits in the cell will arrive at the correct destination. The only concern therefore, is for errors in the data field.

$$B3 = \frac{i}{h+i} \times B$$

$$M3 = \frac{B3}{B} = \frac{i}{h+i} = \frac{1}{1+h/i}$$



The larger the header field, the worse the multiplication effect becomes for a constant information length. The information length chosen is that of an ATM cell, which also has a header length of 5 octets, or 40 bits.

11.3 (15 pts.)

a. We reason as follows. A total of X octets are to be transmitted. This will require a total of $\lceil \frac{X}{L} \rceil$ cells. Each cell consists of (L + H) octets, where L is the number of data field octets and H is the number of header octets. Thus

$$N = \frac{X}{\left\lceil \frac{X}{L} \right\rceil (L + H)}$$

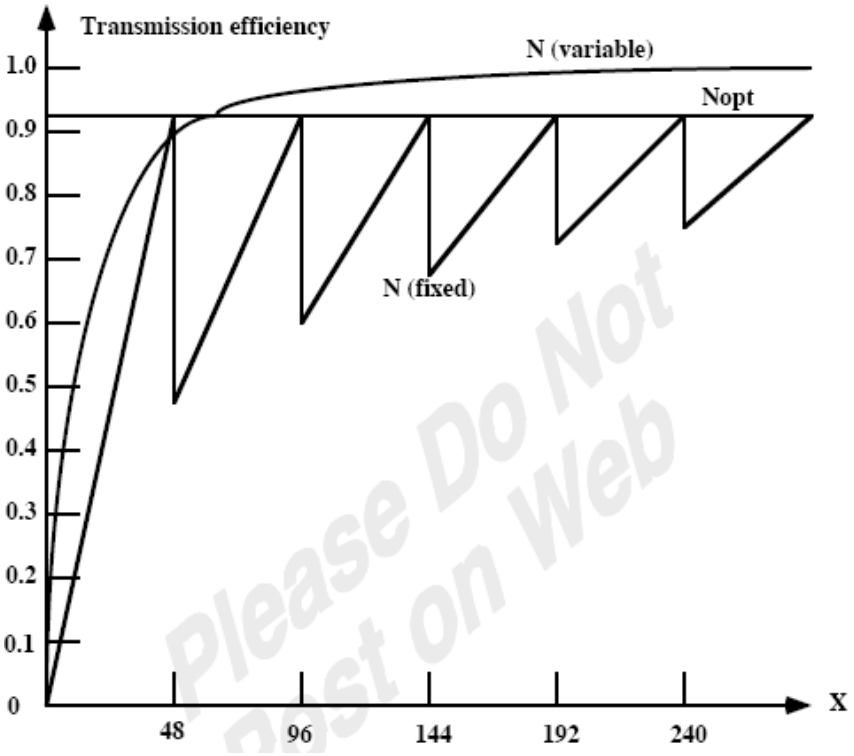
The efficiency is optimal for all values of X which are integer multiples of the cell information size. In the optimal case, the efficiency becomes

$$N_{\text{opt}} = \frac{X}{\left(\frac{X}{L}\right)(L + H)} = \frac{L}{L + H}$$

b. Assume that the entire X octets to be transmitted can fit into a single variable length cell. Then

$$N = \frac{X}{X + H + H_v}$$

c.



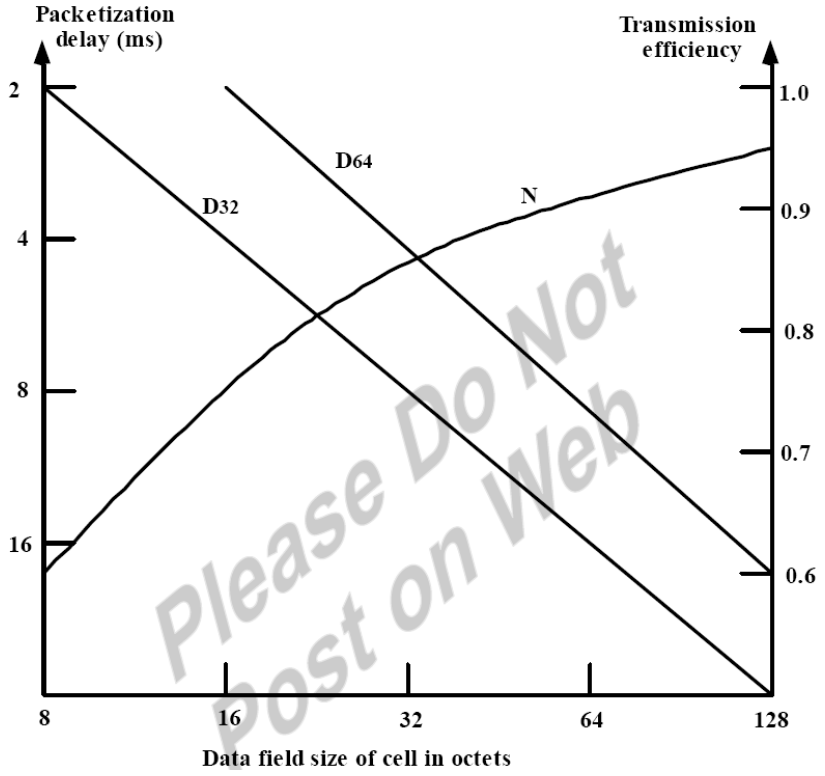
N for fixed-sized cells has a sawtooth shape. For long messages, the optimal achievable efficiency is approached. It is only for very short cells that efficiency is rather low. For variable-length cells, efficiency can be quite high, approaching 100% for large *X*. However, it does not provide significant gains over fixed-length cells for most values of *X*.

11.4 (10pt.)

a. As we have already seen in the preceding problem:

N=L/(L+H)

b. D=8L/R



A data field of 48 octets, which is what is used in ATM, seems to provide a reasonably good tradeoff between the requirements of low delay and high efficiency.

11.5 (12pts.)

a. The transmission time for one cell through one switch is t = (53 x 8)/(43 x 10^6) = 9.86μs.

b. The maximum time from when a typical video cell arrives at the first switch (and possibly waits) until it is finished being transmitted by the 5th and last one is 2 x 5 x 9.86μs = 98.6μs.

c. The average time from the input of the first switch to clearing the fifth is (5 + 0.6 x 5 x0.5) x 9.86μs = 64.09μs.

d. The transmission time is always incurred so the jitter is due only to the waiting for switches to clear. In the first case the maximum jitter is 49.3μs. In the second case the average jitter is 64.09 – 49.3 = 14.79μs.

12.2 (10 pts.) The mean node-node path is twice the mean node-root path. Number the levels of the tree with the root as 1 and the deepest level as N. The path from the root to level N requires N – 1 hops and 0.5 of the nodes are at this level. The path from the root to level N – 1 has 0.25 of the nodes and a length of N – 2 hops. Hence the mean path length, L, is given by

$$L = 0.5 \times (N - 1) + 0.25 \times (N - 2) + 0.125 \times (N - 3) + \ldots$$

$$L = \sum_{i=1}^{\infty} N(0.5)^i - \sum_{i=1}^{\infty} i(0.5)^i = N - 2$$

Thus the mean node-node path is 2N – 4

12.3 (15 pts.)

```
for n := 1 to N do
    begin
        d[n] := ∞;
        p[n] := -1
    end;
d[srce] := 0 {initialize Q to contain srce only 0}
insert srce at the head of Q;
{initialization over }
while Q is not empty do
    begin
        delete the head node j from Q;
            for each link jk that starts at j do
                begin
                    newdist := d[j] + c[j,k];
                    if newdist < d[k] then
                        begin
                            d[k] := newdist;
                            p[nk] := j
                            if k ∉ Q then insert K at the tail of Q;
                        end
                end;
    end;
```

12.4 (12 pts.) This proof is based on one in [BERT92]. Let us claim that
(1) $L(i) \leq L(j)$ for all $i \in T$ and $j \notin T$
(2) For each node j, L(j) is the shortest distance from s to j using paths with all nodes in T except possibly j.

Condition (1) is satisfied initially, and because $w(i, j) \geq 0$ and $L(i) = \min_{j \notin T} L(j)$, it is preserved by the formula in step 3 of the algorithm. Condition (2) then can be shown by induction. It holds initially. Suppose that condition (2) holds at the beginning of some iteration. Let i be the node added to T at that iteration, and let L(k) be the label of each node k at the beginning of the iteration. Condition (2) holds for i = j by the induction hypothesis, and it holds for all $j \notin T$ by condition (1) and

the induction hypothesis. Finally for a node j $\notin$ T U i, consider a path from s to j which is shortest among all those in which all nodes of the path belong to T U i and let L'(j) be the distance. Let k be the last node of this path before node j. Since k is in T $\cup$ i, the length of this path from s to k is L(k). So we have

$L'(j) = \min_{k \notin T \cup i}[w(k, j) + L(k)] = \min[\min_{k \notin T}[w(k, j) + L(k)], w(i, j) + L(i)]$

The induction hypothesis implies that $L(j) = \min_{k \notin T}[w(k, j) + L(k)]$, so we have

$L'(j) = \min[L(j), w(i, j) + L(i)]$

Thus in step 3, L(j) is set to the shortest distance from s to j using paths with all nodes except j belonging to T $\notin$ i.

12.9 (16 pts.)

a. We provide a table for node 1 of network a; the figure is easily generated.

	M	L(2)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	Path
1	{1}	1	1-2	∞	—	4	1-4	∞	—	∞	—
2	{1,2}	1	1-2	4	1-2-3	4	1-4	2	1-2-5	∞	—
3	{1,2,5}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	6	1-2-5-6
4	{1,2,5,3}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6
5	{1,2,5,3,4}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6
6	{1,2,5,3,4,6}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6

b. The table for network b is similar in construction but much larger. Here are the results for node A:

A to B: A-B
A to C: A-B-C
A to D: A-E-G-H-D
A to E: A-E
A to F: A-B-C-F
A to G: A-E-G
A to H: A-E-G-H
A to J: A-B-C-J
A to K: A--E-G-H-D-K

12.10 (10pts.)

h	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	—	∞	—	∞	—	∞	—	∞	—
1	1	1-2	∞	—	4	1-4	∞	—	∞	—
2	1	1-2	4	1-2-3	4	1-4	2	1-2-5	∞	—
3	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	6	1-2-3-6
4	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6

12.12 (12 pts.)

This explanation is taken from [BERT92]. The Floyd-Warshall algorithm iterates on the set of nodes that are allowed as intermediate nodes on the paths. It starts like both Dijkstra's algorithm and the Bellman-Ford algorithm with single arc distances (i.e., no intermediate nodes) as starting estimates of shortest path lengths. It then calculates shortest paths under the constraint that only node 1 can be used as an intermediate node, and then with the constraint that only nodes 1 and 2 can be used, and so forth.

For n = 0, the initialization clearly gives the shortest path lengths subject to the constraint of no intermediate nodes on paths. Now, suppose for a given n, $L_n(i, j)$ in the above algorithm gives the shortest path lengths using nodes 1 to n as intermediate nodes. Then the shortest path length from i to j, allowing nodes 1 to n+1 as possible intermediate nodes, either contains node n+1 on the shortest path or doesn't contain node n+1. For the first case, the constrained shortest path from i to j goes from i to n+1 and then from n+1 to j, giving the length in the final term of the equation in step 2 of the problem. For the second case, the constrained shortest path is the same as the one using nodes 1 to n as possible intermediate nodes, yielding the length of the first term in the equation in step 2 of the problem.

12.16 (8 pts.)

If a node sees a packet arriving on line k from node H with hop count 4, it knows that H is at most four hops away via line k. If its current best route to H is estimated at more than four hops, it marks line k as the choice for traffic to H and records the estimated distance as four hops.

The advantage of this algorithm is that, since it is an isolated technique, minimal node-node cooperation is needed. The disadvantage occurs if a line goes down or is overloaded. The algorithm as described only records improvements, not changes for the worse.