Here is some data obtained from it:

# stations	slot times
5	3.9
10	6.6
20	11.2
40	18.8
100	37.7
200	68.6

52. We alternate N/2 slots of wasted bandwidth with 5 slots of useful bandwidth. The useful fraction is: 5/(N/2 + 5) = 10/(N+10)

53. (a) The program is below (and on the web site). It produced the following output:

λ	# slot times
1	6.39577
1.1	5.78198
1.2	5.36019
1.3	5.05141
1.4	4.84586
1.5	4.69534
1.6	4.58546
1.7	4.50339
1.8	4.45381
1.9	4.43297
2	4.41884

λ	# slot times
2	4.41884
2.1	4.46704
2.2	4.4593
2.3	4.47471
2.4	4.49953
2.5	4.57311
2.6	4.6123
2.7	4.64568
2.8	4.71836
2.9	4.75893
3	4.83325

The minimum occurs at about λ=2; the theoretical value of the minimum is $2e - 1 = 4.43656$.

(b) If the contention period has length C, then the useful fraction is $8/(C+8)$, which is about 64% for $C = 2e - 1$.

```
#include <iostream.h>
#include <stdlib.h>
#include <math.h>

const int RUNCOUNT = 100000;

// X = X(lambda) is our random variable
double X(double lambda) {
    double u;
    do {
        u= double(rand())/RAND_MAX;
    } while (u== 0);
    double val = - log(u)*lambda;
    return val;
}
```

```
double run(double lambda) {
    int i = 0;
    double time = 0;
    double prevtime = -1;
    double nexttime = 0;
    time = X(lambda);
    nexttime = time + X(lambda);
    // while collision:  adjacent times within +/- 1 slot
    while (time - prevtime < 1 || nexttime - time < 1) {
        prevtime = time;
        time = nexttime;
        nexttime += X(lambda);
    }
    return time;
}

void main(int argc, char * argv[]) {
    int i;
    double sum, lambda;
    for (lambda = 1.0; lambda <= 3.01; lambda += 0.1) {
        sum = 0;
        for (i=0; i<RUNCOUNT; i++) sum += run(lambda);
        cout << lambda << "   " << sum/RUNCOUNT << endl;
    }
}
```

54. The sender of a frame normally removes it as the frame comes around again. The sender might either have failed (an orphaned frame), or the frame's source address might be corrupted so the sender doesn't recognize it.

    A monitor station fixes this by setting the monitor bit on the first pass; frames with the bit set (i.e. the corrupted frame, now on its second pass) are removed. The source address doesn't matter at this point.

55. 230m/(2.3×10^8m/sec) = 1 μs; at 16Mbps this is 16 bits. If we assume that each station introduces a minimum of 1 bit of delay, the the five stations add another five bits. So the monitor must add $24-(16+5)=3$ additional bits of delay. At 4Mbps the monitor needs to add $24-(4+5)=15$ more bits.

56. (a) THT/(THT + RingLatency)

    (b) Infinity; we let the station transmit as long as it likes.

    (c) TRT $\leq$ N $\times$ THT + RingLatency

57. At 4 Mbps it takes 2 ms to send a packet. A single active host would transmit for 2000 μs and then be idle for 200 μs as the token went around; this yields