**NW HW1 solution**

1. We will count the transfer as completed when the last data bit arrives at its destination. An alternative interpretation would be to count until the last ACK arrives back at the sender, in which case the time would be half an RTT (25ms) longer.

(a) Total Time = 2 RTT’s (setup) + 1000KB/1.5Mbps (transmit) + RTT/2 (propagation)

= 2.5 RTT + 1000KB/1.5Mbps = 2.5 \* 0.05 s + 1000 \* 8 \* 210 / 1.5 \* 106 s

= 0.125 s + 5.46 s = 5.586 s

(b) To the above we add the time for 999 RTTs (the number of RTTs between when packet 1 arrives and packet 1000 arrives), for a total of 5.586 secs + 49.95 secs =55.536 seconds.

(c) 1000/20 = 50 -> need to send 50 batches;

Total time = 2 RTT (handshake) + 49 RTT (for the first 49 batches) + 0.5 RTT (for the last batch) = 51.5 RTT = 51.5 \* 0.05 s = 2.575 seconds.

(d) Right after the hand shaking is done we send one packet. One RTT after the hand shaking we send two packets. At n RTTs past the initial hand shaking we have sent 1+2+4+···+2n=2n+1 −1 packets. At n=9 we have thus been able to send all 1,000 packets; the last batch arrives 0.5 RTT later. Total time is 2RTT + 9.5RTTs, or 0.575 sec.

2. STDM and FDM both work best for channels with constant and uniform bandwidth requirements. For both mechanisms bandwidth that goes unused by one channel is simply wasted, not available to other channels. Computer communications are bursty and have long idle periods; such usage patterns would magnify this waste.

FDM and STDM also require that channels be allocated (and, for FDM, be assigned bandwidth) well in advance. Again, the connection requirements for computing tend to be too dynamic for this; at the very least, this would pretty much preclude using one channel per connection.

FDM was preferred historically for TV/radio because it is very simple to build receivers; it also supports different channel sizes. STDM was preferred for voice because it makes somewhat more efficient use of the underlying bandwidth of the medium, and because channels with different capacities was not originally an issue.

3. (a) The packet needs to be put on a link twice (by the sending node and the switch) and has to traverse two links. Thus,

Latency = 2\*12,000b/100 Mbps + 2 \* 10 µs = 2\*12,000/(100\*106 ) s + 20 µs = 260 µs

(b) With three switches and four links, the packet needs to be put on a link four times (by the sending node and the 3 switches) and has to traverse four links. Thus,

Latency = 4\*12,000b/100 Mbps + 4 \* 10 µs = 4\*12,000/(100\*106 ) s + 40 µs = 520 µs

(c) Same as in a, except that the switch does not delay the whole packet, but only 200 bits. Thus,

Latency = 12,000b/100 Mbps (node) + 200b/100 Mbps (switch) + 2 \* 10 µs = 12,200/(100\*106 ) s + 20 µs = 142 µs

4. (a) 640×480×3×30×8 bits/sec= 221.2 Mbps

(b) 160×120×1×8×5 bits/sec = 768 Kbps

(c) 650MB/75min= 650×8×220 bits / (75×60 secs) = 1.21 Mbps

(d) 8×10×72×72pixels=414,720bits=51,840Bytes. At 14,400 bits/sec, this would take 28.8 seconds (ignoring overhead for framing and acknowledgments)

5.

a) RTT = 2×distance / speed-of-light = 2 × 55×109 / 3×108 secs ≈ 367 secs

b) data-in-flight on down link = RTT/2 \* link speed = 367/2 × 128 × 103 bits = 23488000 bits = 2936000 Bytes = 2936000 / 220 MB = 2.8 MB

c) Total Delay = Transmission + Propagation = picture size / link speed + RTT/2

= 5 MB / 128 kbps + 367/2 secs = 5 × 8 × 220 / 128 ×103 secs + 367/2 secs ≈ 511 secs

6.

Experimental Results.

7.

Experimental Results.