15-441 Computer Networks (Fall 04)
Homework #1

Chapter 1

5. 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 ($50ms$) longer.

(a) Initial handshaking ($2 * RTT$ = 200ms) + Transmit (1000KB/1.5Mbps) + Propagation ($RTT/2$) $\approx 0.25 + 8$Mbits/1.5Mbps = $0.25 + 5.33$ sec = 5.58 sec.
If we pay more careful attention to when a mega is 10^6 versus 2^{20},
we get $8,192,000$bits / $1,500,000$bits/sec = 5.46sec, for a total delay of 5.71sec.

(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 $5.71 + 99.9 = 105.61$ sec.

(c) Initial handshaking ($2 * RTT$) + Transmit (0) + Propagation ($RTT/2$) + Delay ($49 * RTT$) = $51.5 * RTT = 5.15$ sec

(d) Right after the handshaking is done, we send one packet. One RTT after the handshaking we send two packets. At n RTTs past the initial handshaking, we have sent $1 + 2 + 4 + ... + 2^n = 2^{n+1} - 1$ packets. At $n = 9$, we have thus been able to send all 1000 packets. The last batch arrives $0.5 * RTT$ later. Thus, Initial handshaking ($2 * RTT$) + Delay ($9 * RTT$) + Propagation ($RTT/2$) $= 1.15$ sec.

12. 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.

FDMwas 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

13. 1 Gbps = 10^9 bps, meaning each bit is $\frac{1}{10^9} = 10^{-9}$ sec (1 ns) wide. The length in the wire of such a bit is $1ns * 2.3 * 10^8 m/sec$ = 0.23 m.

15. speed: $3 * 10^8$m/s, distance: $385,000$km, bandwidth: 100Mbps

(a) minimum RTT = 2 (for round-trip) * distance/speed
= $2 * 385,000,000$m / $3 * 10^8$ m/sec = 2.57 sec.

(b) Delay x Bandwidth =2.57 sec * 100Mbps = 257Mbits = 32MB

(c) This represents the amount of data the sender can send before it would be possible to receive a response from the receiver.

(d) We require at least one RTT before the picture could begin arriving at the ground (TCP would take two RTTs). Assuming bandwidth delay only, it would then take 25MB/100Mbps = 200Mb/100Mbps = 2.0 sec to finish sending, for a total time of $2.0 + 2.57 = 4.57$ sec until the last picture bit arrives on earth.

18. bandwidth: 10Mbps, packet size: 5000 bits, propagation delay:10μs

(a) One packet consists of 5000 bits, and so is delayed due to bandwidth = $\frac{5000}{10*10^6}$s = 500 μs along each link. The packet is also delayed 10 μs on each of the two links due to propagation delay, for a total of 1020 μs.

(b) With three switches and four links, the delay is = 4 * 500μs + 4 * 10μs = 2.04ms.

(c) With cutthrough, the switch delays the packet by 200 bits = 20 μs. There is still one 500 μs delay waiting for the last bit, and 20 μs of propagation delay, so the total is 540 μs. To put it another way, the last bit still arrives 500 μs after the first bit; the first bit now faces two link delays and one switch delay but never has to wait for the last bit along the way. With three cut-through switches, the total delay would be: 500 + 1 * 20 + 2 * 10 = 540 μs.

20. (a) The effective bandwidth is 10Mbps; the sender can send data steadily at this rate and the switches simply stream it along the pipeline. We are assuming here that no ACKs are sent, and that the switches can keep up and can buffer at least one packet.

(b) The data packet takes 2.04 ms as in 18(b) above to be delivered; the 400 bit ACKs take 40 μs/link for a total of 4 * 40 μs+ 4 * 10 μs = 200 μsec = 0.20 ms. Hence it takes 2.24ms for a data packet to go from the sender to the receiver, and the corresponding ACK to go from the receiver to the sender. 5000 bits in 2.24ms is about 2.2Mbps, or 280KB/sec.

(c) $\frac{100*6.5*10^8 bytes}{12 hours} = 1.5 MByte/sec = 12$Mbps.

25. 6 point-to-point links, 5 switches n-Byte file, 1 KB packet (24B header + 1000B data)

(a)
- Packet switching: packets pay an ongoing per packet cost of 24 Bytes for a total count of 1024 x (number of packets) = 1024 x $n/1000$ Bytes.
- Circuit switching: circuits pay an up-front penalty of 1024 Bytes being sent on one round trip for a total data count of $2048 + n$ Bytes

So the question really asks how many packet headers does it take to exceed 2048 Bytes, which is 86. Thus, for files $86,000$ Bytes or longer, using packets results in more total data sent on the wire.

(b)
- Packet switching:

Total transfer latency = (transmit delay for all pkts at the source)
+ (per-pkt transmit delay introduced by each switch)
+ (per-pkt processing delays introduced by each switch)
+ (propagation delay for all links)
$= (c * t) + (s * t) + (s * 0.001) + ((s + 1) * 0.002)$
$= 8.192n/b + 8192s/b + 0.003s + 0.002$ sec
$= 0.000002048n + 0.02724$ sec —(1)

where b: bandwidth (=4Mbps $\approx 4 * 10^6$bps)
s: number of switches (=5)
c: packet count (=$n/1000$)
t: per-packet transmit time (=packet size/bandwidth = $1024 * 8$bits/4Mbps)

- Circuit switching:

  Total transfer latency = (transmit delay for the whole file)
  + (propagation delay for the links)
  + (setup cost for the circuit, which is just like
  sending one packet each way on the path (from (1)))
  = (8n/b) + ((s+1)*0.002) + 2*(0.000002048*1000 + 0.02724)
  = 0.000002n + 0.070576 sec —(2)

  Solving the resulting inequality $0.000002048n + 0.02724 > 0.000002n + 0.070576$ for n shows that circuits achieve a lower delay for files larger than or equal to $903,000$B. (n is a multiple of 1000)

(c) Only the payload to overhead ratio size effects the number of bits sent, and there the relationship is simple. The following table show the latency results of varying the parameters by solving for the n where circuits become faster, as above. This table does not show how rapidly the performance diverges; for varying payload size (p) it can be significant.

Given that s: number of switches, b: bandwidth, p: payload size in an 1024B packet,
$n = \frac{0.005s+0.004+2t+s*t}{t/p-8/b}$
$t = 1024 * 8\text{bits}/b$

s	b	p	pivotal n
5	4Mbps	1000B	903000
6	4Mbps	1000B	1050000
7	4Mbps	1000B	1197000
8	4Mbps	1000B	1344000
9	4Mbps	1000B	1491000
10	4Mbps	1000B	1637000
5	1Mbps	1000B	450000
5	2Mbps	1000B	601000
5	8Mbps	1000B	1507000
5	16Mbps	1000B	2716000
5	4Mbps	512B	22000
5	4Mbps	768B	66000
5	4Mbps	1014B	2198000

(d) Many responses are probably reasonable here. The model only considers the network implications, and does not take into account usage of processing or state storage capabilities on the switches. The model also ignores the presence of other traffic or of more complicated topologies.

28. (a) 640 * 480 * 3 * 30 bytes/sec = 27,648,000 bytes/sec = 27,648,000 /(1024 * 1024)MB/sec = 26.4 MB/sec

(b) 160 * 120 * 1 * 5 = 96,000 bytes/sec = 94KB/sec

(c) 650MB/75min = 8.7 MB/min = 148 KB/sec

(d) One pixel requires one bit. Hence, 8 * 10 * 72 * 72 pixels = 414,720 bits = 51,840 bytes. At 14,400 bits/sec, this would take 28.8 seconds (ignoring overhead for framing and acknowledgments).

Chapter 2

5. The stuffed bits (zeros) are in bold:
1101 0111 11**0** 1011 111**0** 1010 1111 1**0**11 0

6. The $\wedge$ marks each position where a stuffed 0 bit was removed. There were no stuffing errors detectable by the receiver; the only such error the receiver could identify would be seven 1's in a row.
1101 0111 11$\wedge$10 1111 1$\wedge$010 1111 1$\wedge$110

Chapter 3

17. (a) When X sends to Z the packet is forwarded on all links; all bridges learn where X is. Y's network interface would see this packet.

(b) When Z sends to X, all bridges already know where X is, so each bridge forwards the packet only on the link towards X, that is, B3→B2→B1→X. Since the packet traverses all bridges, all bridges learn where Z is. Y's network interface would not see the packet as B2 would only forward it on the B1 link.

(c) When Y sends to X, B2 would forward the packet to B1, which in turn forwards it to X. Bridges B2 and B1 thus learn where Y is. B3 and Z never see the packet.

(d) When Z sends to Y, B3 does not know where Y is, and so retransmits on all links; W's network interface would thus see the packet. When the packet arrives at B2, though, it is retransmitted only to Y (and not to B1) as B2 does know where Y is from step (c). All bridges already knew where Z was, from step (b).

21. (a) If the bridge forwards all spanning-tree messages, then the remaining bridges would see networks D,E,F,G,H as a single network. The tree produced would have B2 as root, and would disable the following links:

from B5 to A (the D side of B5 has a direct connection to B2)
from B7 to B
from B6 to either side

(b) If B1 simply drops the messages, then as far as the spanning-tree algorithm is concerned the five networks D-H have no direct connection, and in fact the entire extended LAN is partitioned into two disjoint pieces A-F and G-H. Neither piece has any redundancy alone, so the separate spanning trees that would be created would leave all links active. Since bridge B1 still presumably is forwarding other messages, all the original loops would still exist.

Others

1. 30dB = $10 * \log_{10}(\frac{S}{N}) \Longrightarrow \frac{S}{N} = 1000$.
Bandwidth of telephone line = 3200 Hz.
Hence from Shannon's theorem, maximum speed = $3200 * \log_2(1001) = 31.9$ Kbps.

2. DSL services use a wider range of frequency for transmitting data than what is used in telephone lines for transmitting voice. Hence from Shannon's theorem, it can achieve a much higher transmission speed.

3. Every SONET frame has 3 columns of overhead. Three SONET frames are sent every 125μS on an OC-3 link. Hence total overhead per second = $\frac{3*3*9*8}{125*10^{-6}}$ bps = 5.184Mbps.