Problem 3

Application layer protocols: DNS and HTTP

Transport layer protocols: UDP for DNS; TCP for HTTP

Problem 8

a) $RTT_1 + \cdots + RTT_n + 2RTT_0 + 8 \times 2RTT_0 = 18RTT_0 + RTT_1 + \cdots + RTT_n$

b) $RTT_1 + \cdots + RTT_n + 2RTT_0 + 2 \times 2RTT_0 = 6RTT_0 + RTT_1 + \cdots + RTT_n$

c) Persistent connection with pipelining. This is the default mode of HTTP.

$$RTT_1 + \cdots + RTT_n + 2RTT_0 + RTT_0 = 3RTT_0 + RTT_1 + \cdots + RTT_n$$

Persistent connection without pipelining, without parallel connections.

$$RTT_1 + \cdots + RTT_n + 2RTT_0 + 8RTT_0 = 10RTT_0 + RTT_1 + \cdots + RTT_n$$

Problem 9

a) The time to transmit an object of size L over a link or rate R is L/R. The average time is the average size of the object divided by R:

Δ = (900,000 bits)/(15,000,000 bits/sec) = 0.06 sec

The traffic intensity on the link is given by βΔ =(16 requests/sec)(.06 sec/request) = 0.96. Thus, the average access delay is (0.06 sec)/(1 - 0.96) $\approx$ 1.5 seconds. The total average response time is therefore 1.5 sec + 3 sec = 4.5 sec.

b) The traffic intensity on the access link is reduced by 60% since the 60% of the requests are satisfied within the institutional network. Thus the average access delay is (0.06 sec)/[1 – (0.4)(0.96)] = 0.097 seconds. The response time is approximately zero if the request is satisfied by the cache (which happens with probability 0.6); the average response time is 0.097 sec + 3 sec = 3.097 sec for cache misses (which happens 40% of the time). So the average response time is (0.6)(0 sec) + (0.4)(3.097 sec) = 1.239seconds. Thus the average response time is reduced from 4.5 sec to 1.24 sec.

Problem 16

SMTP uses a line containing only a period to mark the end of a message body.

HTTP uses “Content-Length header field” to indicate the length of a message body.

No, HTTP cannot use the method used by SMTP, because HTTP message could be binary data, whereas in SMTP, the message body must be in 7-bit ASCII format.

Problem 22

For calculating the minimum distribution time for client-server distribution, we use the following formula:

$$D_{cs} = max\ \{NF/u_s,\ F/d_{min}\}$$

Similarly, for calculating the minimum distribution time for P2P distribution, we use the following formula:

$$D_{P2P} = max\{\ F/u_s, F/d_{min}, NF/\left(u_s + \sum_{i=1}^{N} u_i\right)\}$$

Where, F = 20 Gbits = 20 * 1000 Mbits

(although it should be 20*1024Mbits),

u_s = 30 Mbps

$d_{min} = d_i$ = 2 Mbps

Client Server

		N		
		10	**100**	**1000**
U	**300Kbps**	10000	66667	666667
	700Kbps	10000	66667	666667
	2MBps	10000	66667	666667

Peer to Peer

		N		
		10	**100**	**1000**
U	**300Kbps**	10000	33333	60606
	700Kbps	10000	20000	27397
	2MBps	10000	10000	10000

Problem 23

a) Consider a distribution scheme in which the server sends the file to each client, in parallel, at a rate of a rate of u_s/N. Note that this rate is less than each of the client's download rate, since by assumption $u_s/N \leqslant d_{min}$. Thus each client can also receive at rate u_s/N. Since each client receives at rate u_s/N, the time for each client to receive the entire file is $F/(u_s/N) = NF/u_s$. Since all the clients receive the file in NF/u_s, the overall distribution time is also NF/u_s.

b) Consider a distribution scheme in which the server sends the file to each client, in parallel, at a rate of d_{min}. Note that the aggregate rate, N d_{min}, is less than the server's link rate u_s, since by assumption $u_s/N \geqslant d_{min}$. Since each client receives at rate d_{min}, the time for each client to receive the entire file is F/d_{min}. Since all the clients receive the file in this time, the overall distribution time is also F/d_{min}.

Problem 26

Yes. His first claim is possible, as long as there are enough peers staying in the swarm for a long enough time. Bob can always receive data through optimistic unchoking by other peers.

His second claim is also true. He can run a client on each host, let each client "free-ride," and combine the collected chunks from the different hosts into a single file. He can even write a small scheduling program to make the different hosts ask for different chunks of the file. This is actually a kind of Sybil attack in P2P networks.