ECE 358, S'17 — Pencil-n-Paper Assignment 2

Total # Points = 29, Due: Mon, May 29, 11:59:59pm

Instructions Your submission, to the appropriate Dropbox on Learn, must be typeset and in pdf. The Dropbox will be configured to retain the last submission only. One submission per group.

1. *(Adapted from Kurose & Ross, 7e)* Consider a 10-meter link, over which a sender can transmit at 150 bps in each direction. Suppose that packets containing data are $100,000$ bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that if there are N parallel connections, each gets $1/N$ of the link bandwidth. Now consider HTTP, and suppose that each downloaded object is 100 kbits, and that the initial downloaded object contains 10 referenced objects from the same sender.

   (a) (2 points) Are parallel downloads via parallel instances of non-persistent HTTP faster to download all the web objects than serial downloads with non-persistent HTTP?

   (b) (2 points) Does use of persistent HTTP instead provide significant gains over the parallel, non-persistent approach?

   Suppose the link is shared by Bob and four others. Bob uses parallel instances of non-persistent HTTP, and the other four use non-persistent HTTP without parallel downloads.

   (c) (2 points) Do Bob's parallel connections help him get the web objects more quickly than the others? Why or why not?

   (d) (2 points) If all five users use parallel connections with non-persistent HTTP, how does the situation for Bob compare with his situation in Part (c) above?

2. The following experiments are to be on one of the ecelinux machines.

   (a) (2 points) Issue `dig -t A www.ibm.com +recurse` to find the `A` record for `www.ibm.com`. Let the IP address that is returned be <i>. Now issue `dig -x` <i> to perform a reverse-lookup, i.e., retrieve the `PTR` record for <i>. Notice that the `PTR` record is not `www.ibm.com`. Why not? (At most 2 sentences, each of at most 25 words.)

   (b) (3 points) Issue `dig -t A www.ibm.com +norecurse`. Notice that an `A` record is not returned in the ANSWER section. Starting at `www.ibm.com`, and ending at its IP address, <i> from the previous question, write down a table of three columns: *Name*, *Type* and *Value*, which shows the manner in which `www.ibm.com` eventually (recursively) resolves to <i>. So, in this table, the first row should have `www.ibm.com` in the *Name* column. And in the last row of your table, the IP address <i> should be in the *Value* column, with `A` in the *Type* column.

3. *(Adapted from Kurose & Ross, 7e)* Consider distributing a file of F bits to N peers using a client-server architecture. Assume that the server can simultaneously transmit to multiple peers, possibly at different rates, provided the combined rate does not exceed u_s. Let $d_{\text{min}} = \min_i\{d_i\}$, where d_i is the maximum rate at which peer i can download.

   (a) (1 point) Suppose that $u_s/N \leq d_{\text{min}}$. How should the server apportion u_s amongst the clients so that the distribution time is NF/u_s?

   (b) (1 point) Suppose that $u_s/N \geq d_{\text{min}}$. How should the server apportion u_s amongst the clients so that the distribution time is F/d_{min}?

Now consider distributing the same file using the P2P architecture. As in the previous question, simultaneous transmission to multiple recipients is possible. Assume that d_{min} is very large, so that peer-download bandwidth is never a bottleneck.

(c) (2 points) Suppose $u_s \leq (u_s + u_1 + \ldots + u_N)/N$. Devise a distribution scheme that has a distribution time of F/u_s. By "distribution scheme," we mean a specification of who transmits to whom at what time, and how sending bandwidth is apportioned if there are multiple simultaneous recipients.

(d) (2 points) Suppose $u_s \geq (u_s + u_1 + \ldots + u_N)/N$. Devise a distribution scheme that has a distribution time of $NF/(u_s + u_1 + \ldots + u_N)$.

4. (4 points) Consider a Chord circular DHT with 4-bit keys, i.e., each key is in $[0, 15]$. Suppose we generate 5 keys, $k_1, \ldots, k_5$, uniformly at random, independently of one another. What is the expected difference between a key and its nearest neighbour?

   The 'difference' between two keys, k_i, k_j, where $k_i \leq k_j$, is $\min\{k_j - k_i, 16 + k_i - k_j\}$. We write this difference as $k_i \ominus k_j$ and define that to be the same as $k_j \ominus k_i$. By 'nearest neighbour' of a key k_i, we mean one of the 5 keys, k_j, such that $j \neq i$, and $k_i \ominus k_j \leq k_i \ominus k_q$ for all $q \neq i$, $q \in [1, 5]$. For example, given $k_1 = 0, k_2 = 3, k_3 = 10$, $k_1 \ominus k_2 = k_2 \ominus k_1 = 3$, $k_1 \ominus k_3 = k_3 \ominus k_1 = 6$, and $k_2 \ominus k_3 = k_3 \ominus k_2 = 7$. Note that because every key $\in [0, 15]$, the difference between any two keys is $\in [0, 8]$.

5. Consider a Chord circular DHT with m-bit keys. We have n peers.

   (a) (3 points) Suppose every peer p: (i) knows for which keys it is responsible; that is, every peer p has access to a function *itsMe*(k), that outputs *true* if *succ*$(k) = p$, and outputs *false* otherwise, and, (ii) maintains a finger table as discussed in the lectures; that is, maintains a table $\text{FT}_p[i] =$ *succ*$(p + 2^{i-1})$, for all $i \in [1, m]$.

   Write down pseudo-code for an algorithm, call it *lookup*(k), that a peer p executes to determine the next-hop for a query *succ*(k). The output of the algorithm, for any k, is presumably an entry from $\text{FT}_p[\,]$, or p itself.

   (b) (3 points) Suppose it turns out that the keys associated with the peers are equally spaced from one another around the ring. And every peer maintains a finger table as in Part (a) above. By 'equally spaced,' we mean that the following two properties are satisfied. (i) The distance, measured as the number of keys in the ring, between two successive peers is either $\lfloor \frac{2^m}{n} \rfloor$ or $\lceil \frac{2^m}{n} \rceil$. (ii) The sum of all distances between successive peers is 2^m.

   What is the worst-case number of hops to answer a query *succ*(k) for a key k, that is initially issued at some peer p?